TRANSPORTING FABRIC: DECENTRALIZED MULTI-AGENT CONTROL THROUGH A DISTRIBUTED ACTIVE ENVIRONMENT

A PhD Dissertation
Presented to
The Academic Faculty

By

Kyle Motter

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering

Georgia Institute of Technology

December 2019

TRANSPORTING FABRIC: DECENTRALIZED MULTI-AGENT CONTROL THROUGH A DISTRIBUTED ACTIVE ENVIRONMENT

Approved by:

Dr. Wayne Book, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Sundaresan Jayaraman
School of Materials Science and Engineering
Georgia Institute of Technology

Dr. Ling Liu
School of Computer Science
Georgia Institute of Technology

Dr. Anirban Mazumdar
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Tucker Balch
School of Interactive Computing
Georgia Institute of Technology

Date Approved: October 22, 2019

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

The garment manufacturing industry has largely not benefitted from the rapid advances in robotics and automation due to the inherent difficulty in handling flexible materials. At present the vast majority of sewing operations and material handling is still performed by humans in low-wage conditions. However, the industry is undergoing a paradigm shift toward custom and on demand manufacturing, increasing the need for automated handling of cut fabric. This thesis presents a comprehensive system based on novel distributed actuators, called budgers, for fabric manipulation and control.

Using these distributed actuators as a foundation, this thesis explores a system architecture to provide practical, factory-ready local fabric control and a scalable solution for routing material through a large-scale implementation. This is presented within the context of treating the fabric as an "unactuated robot" traversing through the "actuated environment" of a budger array. Examined holistically as applied research, the thesis focuses on the actuation, feedback, and control, necessary for robust fabric manipulation. The budger is physically redesigned for significant performance, manufacturability, and serviceability gains. A custom vision feedback algorithm is presented for real-time stable feedback on the state of the fabric, including position and wrinkle information even in the case of deformation or occlusion. And a system architecture for the unactuated robot provides a scalable solution to handling large numbers of fabric across a decentralized network of these budgers.

CHAPTER 1
INTRODUCTION

1.1 Motivation

This thesis explores automation within the garment industry and the implications of using a novel means of fabric control within the context of a paradigm shift within the industry toward highly customizable "on-demand" manufacturing. This thesis seeks to demonstrate a practical, factory-ready solution to the handling, transport, and manipulation of garment materials through a sewing environment.

1.1.1 Automation in the Garment Industry

Manufacturing has been trending toward automation at an accelerating rate since the introduction of manufacturing machinery. This has mostly taken the form of "hard automation" where fixed, highly customized machines perform rigid, repeatable actions in the production of high volume goods (eg. bottling of Coca-Cola as seen in Figure 1.1). Recent years, with the introduction of robotics, have seen a rapid increase in quickly deployable "flexible automation" in manufacturing within most industries allowing for even greater output with fewer workers. Current economic data for the United States shows the manufacturing workforce has decreased nearly 30% since the mid 1980's, while manufacturing output is at or near an all-time high, almost doubling in that same period [1][2]. The garment industry, however, has not benefited from these advancements and largely lags behind in overall automation.

Figure 1.1: Bottling machinery customized for specific bottle size and shape.

Figure 1.2: Operator loading a dual-head jeans pocket sewing machine which adds decorative sewing and affixes the pocket to the pant leg.

1.1.2 Fabric Handling and Manipulation

Still today, almost every article of clothing or fabric goods is assembled or handled by human operator, whether in the handling and transport of fabric or in the actual sewing of clothing. Sewing operations on garments are mostly done by hand as a person manipulates the tension, feed rate, and control of stacked layers as they pass through a sewing machine. Even in cases where automation has been implemented, one example being an automatic pocket sewer shown in Figure 1.2, an operator often must handle and load the material into a fixed template for every piece. While the continued reliance on manual labor can be partially attributed to the availability of cheap labor pools in global markets, the primary reason is in the inherent difficulty for robots in the handling and control of flexible materials.

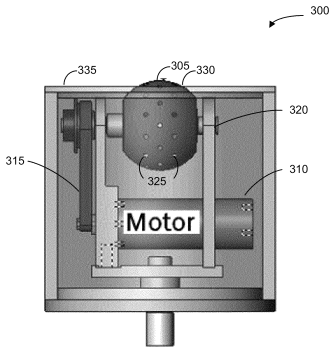


Figure 1.3: Diagram of budger from US patent US8997670B2

Rigid object manipulation is straightforward as the dynamics are easily computed by simple models with contacts and forces. Predicting behavior for flexible materials entails a great deal more computation and feedback about the state of the material than is feasible for typical applications. Even if such computation and feedback could be provided in real time, these results are non-deterministic (material deformation is directly analogous to predicting the location of failure in a beam-buckling analysis and dependent on realistically unobservable properties). As a result, practical implementations in industry rely on carefully constructed hard automation to ensure high probability of success, and accept that there will be some amount of scrap due to errors from fabric fold over, improper placement, etc. In contrast, most robotic interaction with fabric in the research domain attempt to observe and characterize fabric deformations which requires a high degree of complexity in both sensing and manipulation and results in very slow operation. This thesis strikes a balance by relying on some simplifications inherent to the planar sewing operation and builds on the previous work using the patented distributed actuation device (Figure 1.3)[3], referred to as a budger, to allow real time interaction with fabric that can be easily integrated into existing sewing processes. The full budger system is discussed in Chapter 2.

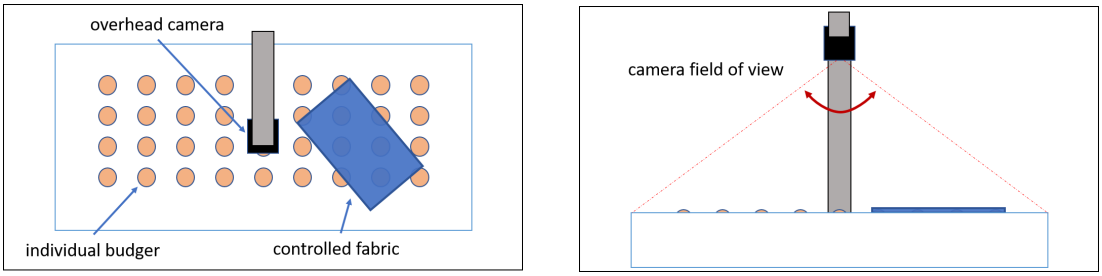
1.1.3 Arbitrary Local Control

The ultimate goal for a single budger table is the continuous ingress and egress of fabric material while maintaining the local (collision free) planning for the fabrics that are on the table. In the most complicated case of fabric handling local to the sewing operation, fabric may need to be kept flat while feeding into the sewing machine, or possibly bunched up to keep the size manageable during rotations while still following a specific sewing edge pattern around a fabric. This motion entails continuous 360 degree positional rotation at the budger-fabric interface, with no time for position corrections. Even under simple point-to-point transport, complications can arise requiring higher complexity motions to properly align the fabric in transit or correct for errors in which portions of the fabric catch a lip and bunch up or fold under. The result is that for general fabric handling, even in simple cases, actuation must be capable of essentially arbitrary control of *at least* the position and orientation and, in all practicality, the flatness of the fabric as well. Four discrete areas needed to achieve this task have been identified.

Feedback. While feedback is needed for closed-loop control, the flexibility of the fabric provides a unique problem. Full state feedback requires a great deal of sensing and any reductions in complexity introduce additional issues such as how to define the position and orientation of an object that can change shape. Any solution must balance these issues to meet the speed requirements of the controller.

Control. The control scheme will need to provide seamless control of the fabric entity by coordinating only the required subset of the budgers in the array. Actual motion of the fabric will be limited to what is feasible for that specific collection of budgers, which will be continuously changing as the fabric moves around the budger environment.

Planning. Because the budger environment is shared among all active fabrics, each fabric must simultaneously follow a collision-free path within the controlled region so that no two fabrics require the same budgers for locomotion. By treating the fabric as the robot,



(a) top (b) side

Figure 1.4: Schematic diagram of integrated budger system for fabric manipulation.

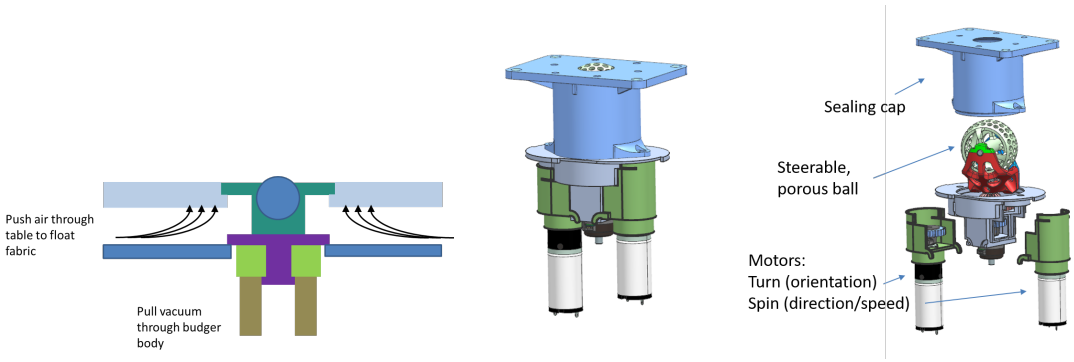
this becomes a mutual multi-agent optimal planning problem.

Scalable Architecture. Each of these problems need to be addressed within the context of a scalable solution that can work from systems as small as a single sewing machine to a large interconnected operation, simultaneously routing and manipulating fabric from many sources to many destinations.

1.1.4 Distributed Actuation

Budger Array as an Environment

This thesis explores the use of distributed actuation for the manipulation and transport of fabric using a patented actuator called a "budger." A budger is a single two degree of freedom actuator (essentially a steerable wheel) embedded in a work surface that allows manipulation of fabric by drawing vacuum through a porous ball to maintain contact with the fabric. The modern design for the budger is shown in Figure 1.5. Performing controlled "rigid body" motion of a fabric requires at least two budgers in contact with the fabric. As the budgers are stationary, these budgers can be arranged in an array (of any pattern) to allow continuous multi-budger contact as the fabric is transported. Figure 1.4 shows a diagram of a complete budger system, with an overhead camera for feedback and array of budgers embedded into a work surface. This collection is a unit area of actuation and referred to as a "budger table" (for obvious reasons). The budger system was invented to fill a niche role in the larger context of a fully automated sewing workcell to provide



(a) diagram of airflow (b) modern budger design (c) exploded view

Figure 1.5: Overview of the modern budger design.

both general transport and gross management of the fabric during the sewing operation. In contrast to other technologies that seek to handle fabric in a manner similar to human hands in 3D space, the budgers take advantage of the inherent planar motion of the majority of fabric handling. Planar motion allows the actuators to be embedded into the table, which frees up the overhead space for manual manipulation and interaction with human workers and provides an unobstructed view of the workspace by the overhead camera for feedback. The distributed nature of the budger system is inherently scalable, allows for a "mechanical multi-tasking" (such that a single surface can handle nearly as many pieces of fabric that can fit on it - in contrast to a gantry system which can handle one at a time), and provides available over-actuation, allowing for the possibility to control additional degrees of freedom of the fabric state (such as wrinkle characteristics).

Previous work with these budgers used a small array of 6 prototype budgers in open loop control in which all budgers were controlled as a single grouped entity to apply linear or arc motion of a fabric as proof-of-concept. Current work is focused on transitioning to a practical, factory-ready implementation, increasing the array to a size large enough to handle multiple fabrics. This requires a scalable software architecture and the introduction of robust feedback on fabric state for both closed-loop control and proper selection and partitioning of the budgers in order to handle multiple pieces of fabric.

1.2 Contributions and Scope

The work presented in this thesis is primarily applied research into distributed actuation for use in industrial fabric handling. It is positioned in an industry that is rapidly growing toward a workflow that deals with large quantities of various shapes of single layer fabrics for custom garment manufacturing, filling the niche between the existing research in low-speed complex garment handling and high speed mass-production processes. This work focuses on using and expanding upon many existing technologies to integrate into a single system to provide new functionality.

These contributions include:

- Providing design enhancements to a novel distributed actuator (budger) to make the system viable in real-world industry conditions.
- An extension of the iterative closest point (ICP) algorithm for fast fabric position tracking using basic shape information that remains stable under common fabric deformations.
- Introducing a combination of fast wrinkle detection from the depth image with a distributed control algorithm for removing and/or controlling wrinkle state while the fabric is in motion.
- The combination of each of these pieces into an architecture of vision, actuation, and control to generate a novel, fully functional fabric transportation system through distributed actuation.

1.3 Outline of Thesis

The following thesis is broken into three chapters, actuation (Chapter 2), sensing (Chapter 3), and control (Chapter 4), corresponding to the main requirements of a robotic system. The first discusses the unique characteristics and requirements of the budger to allow for

fabric transport and the improvements made to meet these requirements. For sensing, the chapter focuses on the needs for real time tracking of a flexible material with a priori shape information and the hardware and algorithms developed to handle these challenges. Finally with control, the chapter takes the work presented in the preceding chapters to create a unified system for the scalable control of the budgers within the context of an unactuated robot and an actuated environment. Throughout, there will be a theme of modularity and speed tying back each part to the overall goal of providing research that results in a functioning high-throughput fabric manipulation system that is ready for the demands of emerging industry processes.

CHAPTER 2
DESIGN IMPROVEMENTS FOR DISTRIBUTED FABRIC ACTUATION

2.1 Motivation

As mentioned in the introduction (Section 1.1.3), the goal is to completely control a fabric entity (piece of cut cloth to be manipulated, further referred to interchangeably as ”fabric”) during conveyance which requires sufficient actuation authority. This actuation is to be provided by the budger.

2.1.1 Familiarity with the Budger Concept

While this thesis does not include the inception of the budger, presented here is a summary of the origins of why the budger was developed, what problems it solved, and what mitigation options exist to deal with any challenges it brings.

Origin and Benefits

Existing fabric-interaction robotics use “overhead” mechanics (see Section 2.2) to handle by either grasping and draping or stamping down with a plate (generally speaking). This blocks the area from interaction with other mechanisms or humans as well as obstructs the view of the workspace for any visual feedback. The budger was developed as a means to address this concern, allowing for fabric conveyance and manipulation while still leaving the area free to be supplementally interacted with. The resulting distributed actuation design provided additional follow-on benefits. The throughput per unit area is increased relative to traditional robotics. Whereas with a gantry or robot arm, for a given robot workspace, only one fabric can be handled at a time, the distributed budger system can manipulate as many fabrics as can be contained in that area (allowing for certain constraints). The

distributed nature also provides inherent robustness against failure. Although the higher number of actuators increases the likelihood of any individual failure to occur, the likelihood of any failure to shut down a region of operation is decreased and the failed part can be avoided and replaced without interrupting flow. This higher number of actuators also provides a higher degree of controllability of the fabric state. With each budger having 2 degrees of freedom, when 2 or more budgers are in contact with the fabric, it's *possible*, though not guaranteed, that additional fabric degrees of freedom (such as wrinkle state) can be controlled at no additional actuation cost. (It is essentially a "free" side effect of the over-actuation from distributed actuators.)

Mitigation of Known and Expected Issues

Most design choices bring with them trade-offs and the budger is no exception. Presented here are some of the known issues of utilizing the budgers in a fabric manipulation system and referenced within this thesis are options to mitigate, alleviate, or otherwise handle these expected deficiencies.

First and foremost, the question of any distributed system is the required density for effective operation. The higher the density, the less practical the design choice may be as the cost and physical complexity of the system increases which may render it infeasible or impractical. The required density of the budgers is primarily dictated by (1) the rigidity or relative compliance of the fabric, (2) the effective friction between the fabric and the rest of the table surface (non-actuated portion) and (3) the size of the fabric, with smaller fabric pieces needing a tighter budger pattern. This is provided for in the following three ways. Inherent to the control (Sections 4.4 and 4.5) and planning (Section 4.6) design, irregular budger patterns are allowed, which in turn allows for there to be sections of higher density "lanes" (discussed in Section 4.3.2) which can minimize the number of local budgers needed for a given operation. The global (intra-warehouse) number of budgers can also be limited, by locating budgers in "zones" where complex conveyance and control is required

and allowing for simpler conveyor belt transportation elsewhere, provided for by a global routing and handoff system (also Section 4.3.2). Lastly, table-based friction is reduced to near zero by providing an air-table surface (Section 2.3.4).

Fabric rigidity comes up again with regard to maintaining flatness and fabric shape. In the case of a standard over-head robot mechanism (gantry/arm), a plate stamps down on the fabric to hold it while in motion, which by default also maintains the fabric's planar rigidity. While this can be considered a benefit of the budgers since the over-head robot requires a plate as large or larger than the largest piece of fabric expected to be transported and the budgers can simultaneously handle various fabric sizes (mentioned above), the lack of out-of-plane stiffness presents a problem. There is no physical mechanism keeping the fabric rigid, and any unexpected snag or improper motion can and will cause the fabric to buckle, which is mitigated with wrinkle control (Section 4.5.1) to provide "virtual stiffness".

2.1.2 Status of Preceding Research

As mentioned, the budger as an actuation paradigm for fabric control pre-dates and is directly extended by the work presented in this thesis. The status of the budger system at that point was limited to early prototyping/proof-of-concept. The primary purpose of this early design was to validate the actuator concept with the ability to successfully contact and move fabric with a table-embedded actuator, and as such, it presented some inherent drawbacks. Although just mentioned here, they will be further expounded upon and discussed relative to the changes in Section 2.3. Inherent to the design, which was a direct evolution of the function, the motors were embedded internal to the actuator and inline with the rotation axes resulting in physically limited range of motion for the steering/pointing direction of the ball, as well as performance deficiencies related to motor sizing required to package motors internal to the actuator. While initially one budger was used to show fabric movement, this was expanded to a prototype budger table consisting of six independent budgers. All of these budgers (although individually independent and controllable) were program-

matically treated as a single unit. This was done without any kind of feedback/sensing with the fabric itself, so each budger (and subsequently the fabric) is controlled in open loop, providing fixed translation or rotation about a point as just a single demonstratable operation as a given time. Further discussion of the control changes are discussed in Chapter 4.

2.1.3 Anatomical Overview

For those unfamiliar with the budger itself, the following serves to cover the fundamentals of the budger actuator as a lead in to the work on changes and improvements. With the primary purpose of budger system to free up the overhead above the fabric traversal workspace, the actuators need to be embedded within the work surface which brings certain constraints driving the budger design. Controlling any object required interaction with a directable force. For the budgers, that is accomplished via an "upside-down" steerable wheel, that interacts with the fabric (as if you flip a car on its back and drive the road around with the wheels). As such the fabric is manipulated under non-holonomic control.

This is accomplished via three primary components of the budger concept:

Motors. The budger is a two degree of freedom actuator - it needs to steer and spin. To provide the motion it needs two independent sources of motion, which were stepper motors in the original design but can be generalized to any rotational motor (through preferably direct current based electric motors for compactness and ease of use).

Vacuum. The actuation provided by the motors requires some normal force between the fabric and the budger; however, because the fabric's out of plane stiffness is so low (essentially zero) and localized weight so low, the any friction between the budger and fabric causes the fabric to lift off the budger rather than translate. Vacuum is used to provide the normal force by drawing air through the fabric at the fabric-budger interface and creating the necessary friction. This requires a sealed-chamber design in which vacuum can be pulled through with the only porous region being the driving wheel.

Ball. The steerable wheel of the budger is actually a truncated sphere. With the vacuum requiring a seal between the driving wheel and budger housing to pull air through the wheel itself. However, the wheel still has to turn on its vertical axis, leaving room for air to escape. The solution is to use a porous spherical wheel (the result of revolving a wheel 360 degrees about its vertical or y-axis) to provide a constant seal as the ball turns.

With regard to design moving forward, these are treated as core aspects of the budger and fixed as part of this actuation paradigm.

2.1.4 Targets for Performance Enhancement

So with this understanding of the existing budger and the fundamentals left unchanged, discussed here are aspects available to alter or improve along with some desired performance goals for the redesign effort.

Harder. In terms of durability and serviceablility, the final budger should be a feasible and industry-ready actuator. This entails both enhancement of maintenance-free service life and the ability to quickly replace or exchange parts in the case of failures to keep the full system operational. This goal does not have hard numerical targets.

Better. While there are inherent limits to a non-holonomic actuator, the general design can be improved to allow for continuous rotation of the turn (orientation) of the ball while still maintaining spin to provide more seamless control. This is a critical requirement of complex motion and a "must-have" feature.

Faster. Both the turn rate and top spin speed need to increase to be industry-capable. The spin speed correlates to the max speed of the fabric as it passes by, which is pegged to the expected speed of a sewing operation at 0.5m/s. Turn speed would ideally be infinite. In practice, the rate of turn as fabric passes by is fairly low, but the turn speed also determines the "readiness" rate of a budger. Between fabrics or when not in use, the rate at which a budger can switch to the desired orientation, determines how far ahead planning needs to

select budgers to use. This readiness time should be as low as possible and is targeted for sub-1-second times, under 300ms.

Stronger. While the strength of a motor relative to the weight of a fabric should be no issue, it was possible to back-drive the stepper motors in the original design and miss steps. It is important to ensure that the motor will *not* skip any commanded motion by losing steps and lose its known position. While this obviously deals with motor torque, it also includes a good friction connection between the budger and the fabric to validate the "no-slip" assumption used for control calculations.

Smarter. The budgers have sufficient local computational power to provide additional capability in their own control computation. As some of the controls for the budger are altered or improved (due to other changes in the design), these controls should be locally computed to keep the fabric controller free to give simple, high-level commands.

Smaller. One of the biggest drivers of capability for the budger system in terms of size and type of fabric that it can handle is the density of budger placement. While it is not necessary to maintain a highly dense pattern *everywhere*, it is important to have the capability to place budgers close together. This is completely dependent on the footprint of the device itself, and the aim is to provide a design that allows budger contact points to be placed within 100mm of each other.

2.2 Related Work

2.2.1 Automated Fabric Handling

Much of the current research in fabric handling robotics focuses on either machine learning for identifying and predicting various aspects of the fabric state and/or using a low number of touch points to manipulate a fabric, usually with serial arms and grippers [4] [5] [6] [7] [8] . The closest analogue to planar fabric control is the work by Johannes Schrimpf related to manipulating multiple pieces of fabric during the sewing operation [9] [10] .

These approaches are either too slow for realtime feedback, or specific and localized around the sewing operation. None address the more general problem of fabric handling for traversal. The presented system of distributed actuation, a direct extension of initial proof-of-concept work with the budgers [11], remains unique in this industry.

2.2.2 Distributed Actuation

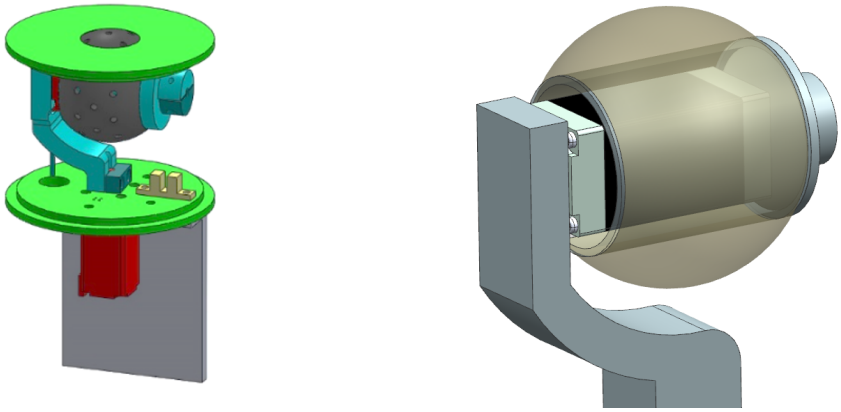
While there is little to no research into fabric handling with distributed actuation, the dynamics and control of rigid bodies on macro scale distributed manipulators (somewhat analogous to the budger system) is well studied, dating back to the late 90's. Luntz's work introduces the concept of a virtual vehicle very similar to the "unactuated robot" presented here [12]. This approach relies on slip (or stick/slip) at the interface between the conveyed object and counter-rotating manipulators, generating a known force imbalance to predict dynamics of the object[13] [14] [15] [16]. While interesting, this requires that the "vehicle" is rigid which obviously does not apply to transporting fabric.

2.3 Significant Changes to Budger Design

While the performance targets are laid out in Section 2.1.4 delineated by the type of target, each design change has the possibility of multiple benefits. This section examines the individual changes to the design and relate them back to their impact on the performance, with a full report on the performance enhancement and how it compares to the target goals covered in Section 2.4.

2.3.1 Moving Motors Outboard of Central Housing

The single highest impact change to the design, from which all others follows is the decision to move the motors outboard of the budger housing. As you can see in Figure 2.1, in the original design, the motors were integrated within the axes of rotation. This design choice presents four primary drawbacks:



(a) Original budger overall design.

(b) Closeup of exposed ball with integrated stepper.

Figure 2.1: Original budger design with (2.1a) sealing surface and stepper motors colored in red for visibility and (2.1b) closeup view of stepper integrated into the ball (shown translucent for visibility)

- Keeping the motor inside the ball (Figure 2.1b) adds a significant amount of weight to the ball and results in a high moment of inertia for the turning axis (vertical).
- The integrated motor is the primary driver of the rotational limit as the wires leading to the motor can not be infinitely wound around the axis.
- The mechanical needs of integrating the motors in this way tightly couples the motors with the shafts. This makes removal of the motors in the case of failure extremely difficult and tedious, requiring deep access and complete disassembly of the budger.
- Motor size is completely dependent on the desired size of the budger (and vice versa). A small ball necessitates a small motor, which adversely impacts the performance desires for torque and speed.

Moving these motors out of the ball and housing simultaneously improves on a number of the target metrics and remove a great deal of constraint on the motor choices.

Removing the motors from direct drive of the turn and spin necessitates some kind of belt or gearing to transmit power. To fulfill the desire for a small footprint, this power

transmission system needs to be as tightly packed as possible. The following sections discuss the specifics of designing such a system to meet all of the design goals.

Serviceability

With the goal of keeping the motors easily removable, they were placed as far to the outside of the housing as possible with a quick release mount (show picture relating to old model). For the motors to actually be able to be quickly released, a gear interface was used for power transmission since the motors could then simply be removed and replaced directly meshing with the internal gearing of the budger without the need for any tooling.

Packaging

To keep the footprint as small as possible, it's desirable to orient *both* motors vertically with the budger. This allows the greatest freedom in motor sizing choices under the assumption that motor diameters are much lower than motor lengths. And by freeing up the motor to grow in length (essentially arbitrarily), motor torque can be easily adjusted with attachable planetary gears (see Section 2.3.2).

The tight packaging requirement also severely constrains the options for gear design, particularly in the case of a completely custom setup. To combat cost and complexity, it is helpful to keep the total number of parts low, which means also keeping the total number of gear meshes and consequently the opportunity for gear reductions/increases low.

Continuous Rotation

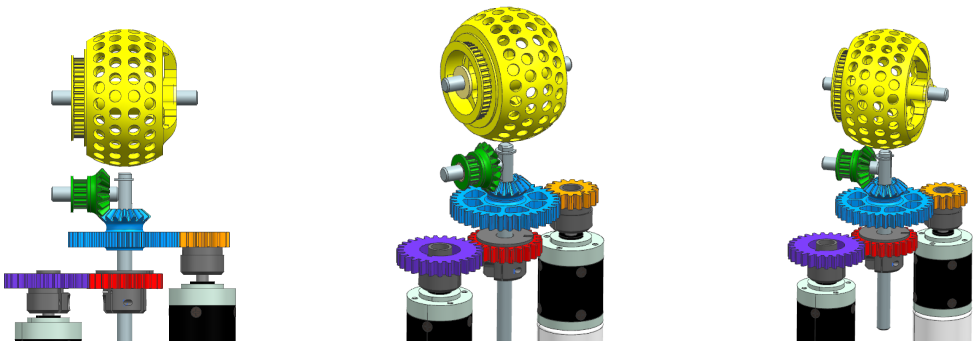
Along with these other constraints, the gearing system must be fully compatible with a continuously rotating spindle. The power transmission has to be able to pass through from the outboarded motor to the continuously rotating spindle to provide a spin to the ball as well.

Gearing Design

Desired Mechanics For packaging, the motors are oriented vertically, so the transmission axis starts vertical, which is perfect for the turning (orientation) of the budger, but need to converted to horizontal spin for the ball speed. To maintain serviceability, the motors should be easily separable and not integrated with the drivetrain (i.e. no shaft couplings), requiring a spur gear interface. Finally, to allow for continuous rotation, the motion transmission for the spin motor must operate continuously, regardless of budger position, necessitating a coupling akin to a differential drive (similar, but not the same).

This configuration shown in Figure 2.2 packs the two vertical motors as tightly along the center turn axis of the budger as possible while maintaining easy separation by simply pulling them away radially. The gear train following the motors places the rotation and spin drive inline with each other on the same axis, further tightening the packaging, with the additional benefit of limiting the vacuum loss from the ball housing region to a single point at the axel bushing. The portion inside the ball housing uses a custom spur-miter-miter-belt drive to convert one of the two concentric spinning motions to horizontal drive for the ball.

Part Sourcing One particular challenge in creating this design is the part sourcing for the internal gears. No off-the-shelf parts existed for such a mechanism, and the needs for gears are very specific, with sets of dissimilar gears needing to be fused together to keep the packaging as tight as possible. One vendor quoted producing the desired gears between $30 and $100 *per* gear, even with relatively high quantities (1000 pieces or more). As an alternative, existing solid models of small gears were used to generate custom combinations which could be 3D printed. However, this introduces an additional design constraint on the complexity and size of the gears. In order to be reasonably printed, the teeth need to be fairly large (2mm or greater spacing between teeth) which puts a lower bound on the total size of the gears themselves, which has to be accounted for in the design. These custom

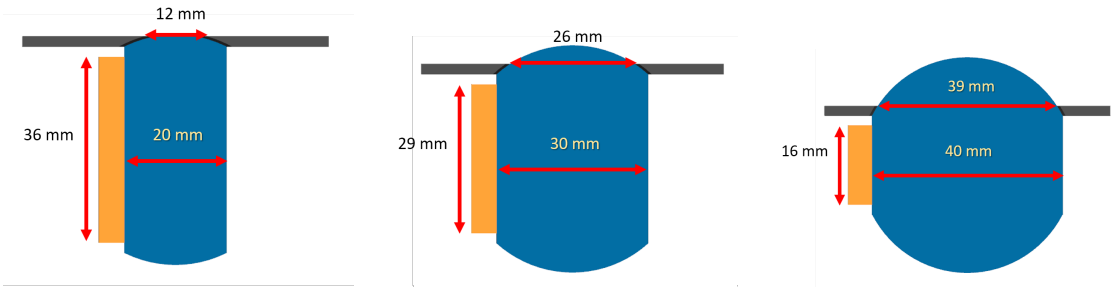


(a) (b) (c)

Figure 2.2: Multiple views of the internal budger gearing with each gear color coded (belt connecting the green miter and yellow ball not shown). [Purple: turn spur gear; red: turn shaft gear; orange: spin spur gear; blue: spin shaft miter gear; green: spin miter-belt drive; yellow: ball belt drive]

gears are able to be produced for less than $3 each from commercial 3D printers, which makes these gears anywhere at least an order of magnitude cheaper.

Gear Sizing Since the spindle rotates around as the center shaft turns the pointing direction and the miter gear driving the spin of the ball rides on top of center shaft, the two actions are coupled. In practice, the result is that the maximum speed of the ball (as governed by the maximum speed of the spinning drive motor) is altered depending on rate of change of the budger orientation. Likewise, a more pressing problem in the positioning of a fabric, is that the ball will precess either forward or backward as the budger turns (changes orientation) while intending to keep the fabric stationary (no spin). The precession rate is completely determined by the two gear interfaces shown in Figure 2.2. These gear combinations are chosen in order to minimize the amount of coupling and reduce the perceived precession, while obviously maintaining a feasible package. Any remaining disturbance introduced by the precession can be eliminated with the closed-loop control discussed in Sections 3 and 4. Given the limitation of sourcing parts and extremely tight space allowance, the range of options almost exclusively relied on the belt-drive ratio between the miter-belt and the ball-belt diameters as the miter-gear mesh had to be 1:1. Choosing the

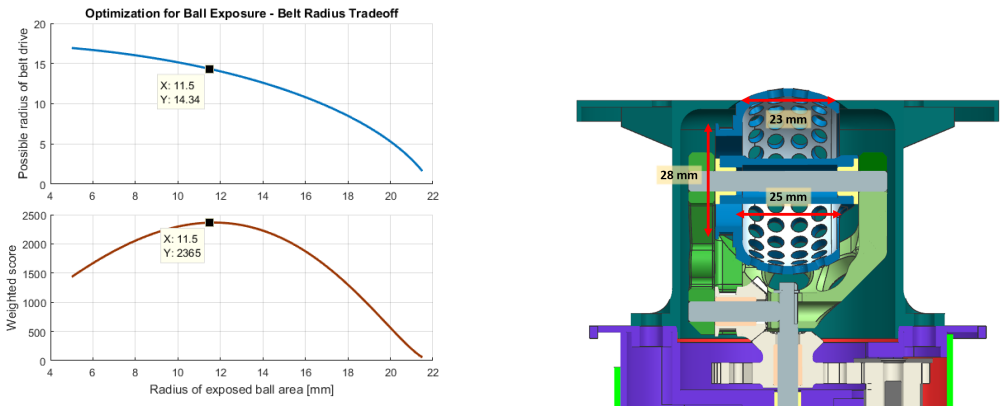


(a) (b) (c)

Figure 2.3: Illustration of tradeoff between ball exposure area and diameter of belt drive.

diameter of the belt drive on the ball comes down to a tradeoff between the overall size and exposed surface of the ball and the diameter of the belt drive assuming some minimum clearance between the belt and the budger housing. This tradeoff can be seen more clearly in Figure 2.3 with different ball sizes and how that impacts the belt diameter (in orange) and exposed surface. Figure 2.4a presents a calculated model of the tradeoff between the diameters, using a cap thickness of 2mm and belt clearance of 3mm. Although these design choices are somewhat arbitrary and balancing a handful of issues, by generating a cost function more heavily weighting the belt diameter (since it's exclusively the driver of the precession rate), with a function of $d_{ball} * d_{belt}^2$ (where d_{ball} is the exposed ball diameter and d_{belt} is the diameter of the belt drive attached to the ball), the ball exposure is "optimal" in the 11 to 12mm radius range. Using this as a guide, you can see these numbers approximately incorporated into the final budger design in Figure 2.4b, once sized against available belt drive models.

Conversely, the diameter of the belt-miter gear was minimized using the smallest available size that would still fit on the shaft. The sizing resulted a ratio of 26.7:11.5 from ball to miter and a 0.43 precession rate of the ball. That is for each 360 degree turn of the spindle, the ball rotates forward 0.43 turns. With a ball diameter of 45 mm, the ball will precess less than 2 cm per revolution of the spindle. This can be further improved as, in practice, a budger will only need to turn a maximum of 180 degrees while in continuous contact with fabric under the most extreme expected fabric maneuvers, bringing the fabric error down to



(a) (b)

Figure 2.4: Computation of "optimal" belt diameter with final sizing in actual design.

under 1 cm. In addition, this rotation typically occurs at a slow speed as a fabric traverses the table reducing the impact. This remaining 1 cm of positional error, while small, still has the potential to generate a mismatch of speeds between budgers and the generation of a wrinkle within the material. Using the wrinkle-feedback (Section 3.4) this is mitigated with the wrinkle-control algorithms (Section 4.5).

With the upper gears sizing set exclusively by the tradeoffs for cross-talk between motors, the speeds and torques for each degree of freedom are set by the motors themselves and the gear ratio between them and the spin-drive gear and the turn-drive gear (purple-red and orange-blue meshes in Figure 2.2). One constraint on this gear selection is the radial distance from the central turn shaft to the motor shafts and the maximum gear size, as these are single-interface gears, so a large gear ratio necessitates a physically larger gear. Moving the motors outboard frees up the choice in motor to meet any desired specifications, so these gear interfaces could be relatively arbitrary. However, as discussed in Section 2.3.2 in sourcing the motors, for uniformity the same base motor was chosen, but were limited in the choice of attached planetary gear. The speeds governing gear choices, then, are set by the desired speeds of the ball and turn shaft relative to the speed of the nominal motor *and* the maximum distance between shafts and footprint size. For these interfaces, the chosen ratios were 17.5:8 for the spin drive with a 3.7:1 planetary gearbox and 11.5:13.5 for the

turn drive with a 19:1 planetary gearbox, for a total resulting ratio of about 8:1 and 16:1 respectively, resulting in a fabric travel of 0.56m/s given by the 45mm diameter of the ball.

2.3.2 Selecting Higher Performance Motors

Switch to Brushless DC

By outboarding the motors, the size and shape of the drive motors was no longer a hard constraint on the design aside from the desire to continue to keep the footprint as small as possible. This change, along with orienting the motors vertically, gave us the freedom to match motors to the desired performance parameters, rather than packaging.

The original budgers were driven by high-torque stepper motors drawing 0.8A at 24V and max torque of 4.2oz-in. In practice this resulted in a max spin speed of 75RPM of the ball and a point to point turning time of over 1 second on average (as low as 0.7s for a 45 degree turn, and as high as 1.8 seconds for 180 degree turn). This performance is partially due to the direct drive and resultingly low torque which made it easy for the motors to skip steps. In addition to poor performance, the motors would draw a constant 0.8A even while stationary. While this is normally a "feature" of stepper motors and their high stationary holding torque, in the case of the budgers, which may not be driving cloth a majority of the time, it means a significant excess power consumption. Anecdotally, this first presented itself as a problem when leaving the budgers on for a long period of time. These little motors sitting idle drew enough power to heat up metal baseplate which, in turn, heated up the clamps holding them in place. The clamps eventually expanded under the heating (they were HOT) and budgers spontaneously fell out of the table.

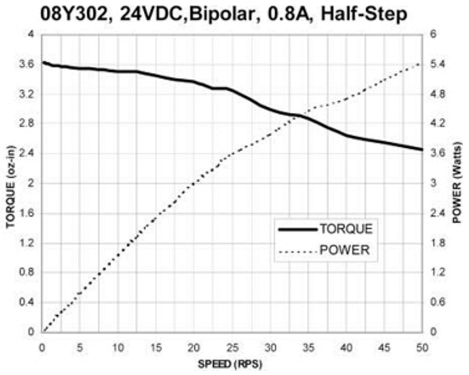
These characteristics drove us toward brushless DC motors. A stepper motor is essentially an open-loop position controller, whereas brushless DC motors are like closed-loop velocity controllers. Because feedback on the fabric position will be implemented (Chapter 3), a velocity controller can be built (Chapter 4) that can forgo the local benefits of a steppers positioning abilities. Constructed similarly to steppers, brushless DC motors offer

high lifetime and quiet operation, as well as having no current draw while not in use.

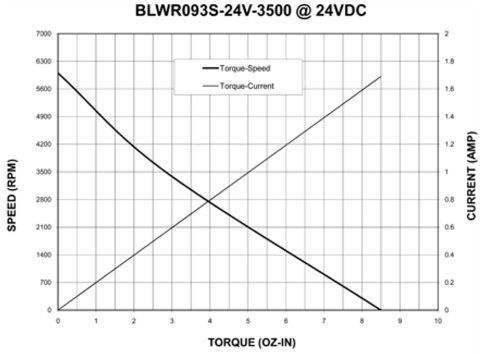
Performance Characteristics

With length of motor no longer being a problem, it was possible to find small diameter (NEMA 8) motors but at a much longer length to achieve the desired power rating and freedom to add planetary gearboxes to tune the speed accordingly to interface with the gears designed in Section 2.3.1.

Figure 2.5 shows representative performance curves for both a stepper motor and brushless DC motor. The primary takeaway is the difference in torque curve and current consumption. The stepper motor draws a nearly constant 0.8A while powered on which provides the high torque across the full range of speed, while the relationship between torque-speed and torque-current are inverse for the brushless motor. Accordingly, the brushless motor can run at higher speed with lower current draw if the budger doesn't require full torque. The original budger used two NEMA 8 stepper motors drawing 0.8A at 24V and a torque output between 2.4 and 3.6 oz-in. With a persistent current draw, that is an average power consumption of 38.4 Watts. In switching to brushless DC of comparable power output, the new budger uses 12V gear motors with a rated torque of 2.8 oz-in at 1.12A (and peak torque of up to 8.5 oz-in). Right off the bat, if both motors ran at nominal capacity, full time, the power consumption is 26.9 Watts, a savings of 30%. However, under some conservative estimates for budger usage this improves significantly. In fact, the budger table is driven by three 12V/10A power supplies, so it must average less than 12W per budger. In any practical scenario the consumption is likely far less due to the budgers requiring full torque during acceleration or quick direction changes. In general, fast direction changes are at the start and end of budger activation and holding torque and drive torque should be greatly reduced during fabric traversal. Under the assumption that at any one time only 50% of the budgers in the full system will be in use and that those 50% will only be using 50% of the rated torque (conservative average), the power consumption drops to



(a) Power/torque curve for stepper motor in original budger



(b) Available torque/power curve for 24v version of selected brushless motor (for reference)

Figure 2.5: Representative performance curves for comparable stepper and brushless motors.

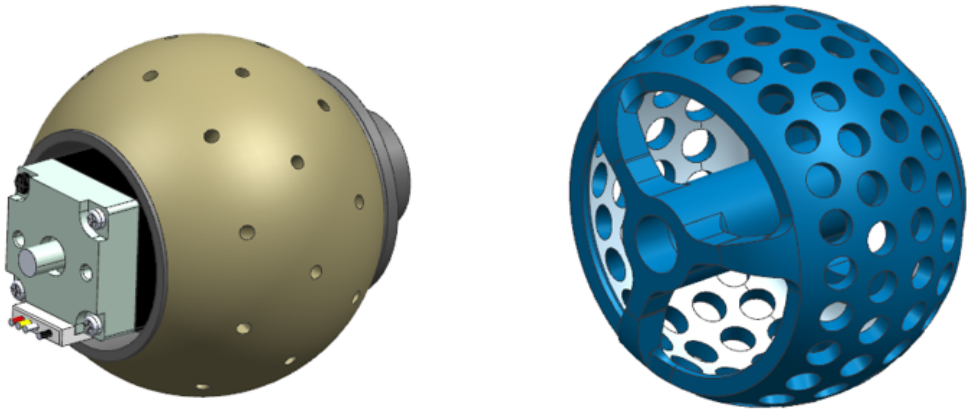
6.7W, resulting in a total power savings of over 80%.

2.3.3 Streamline Ball for Weight and Airflow

Removing the motor from the ball and adding internal gearing required a complete redesign of the upper stage of the budger, including the spindle and ball to account for the new mechanics. This change was a good opportunity to improve the weight distribution and porosity of the ball for air flow as well.

Modification to Structure

The original ball started as a solid silicon sphere, which was machined to remove it's core, fitted on an aluminum sleeve and then holes drilled radially to allow air to pass through. This machining was a delicate process and the air holes were small to keep the ball from breaking apart. Figure 2.6a shows this original ball structure (with stepper motor inside) with the new ball in Figure 2.6b for comparison. The new structure was a product of two primary goals: integrating the belt drive gear (as discussed in Section 2.3.1) and reducing the overall weight and moment of inertia. Reducing the inertia was accomplished by turning the entire ball into a shell surface with very few supports. The rigidity of the nylon



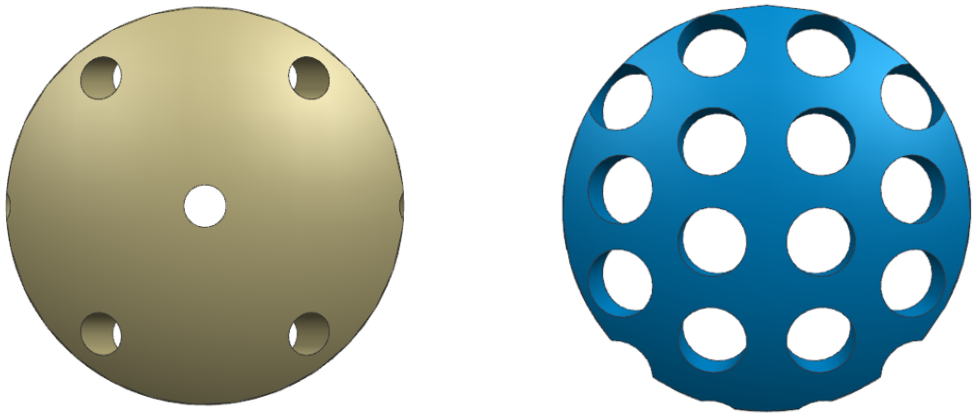
(a) (b)

Figure 2.6: Comparison of original and new budger ball design, showing structure, porosity, and internal air-flow volume.

printing material allowed for a wall thickness of 2mm. Adding the belt drive to the ball, required some more tradeoffs between the exposed area of the ball and the allowable radius of the belt drive (which is desired to be as large as possible). Figure 2.3 shows the tradeoff space for the belt drive: as the exposed area of the ball increases, the belt drive gear diameter must decrease. Together the combined finished ball resulted in a rotational moment of inertia about the spin axis of 2.67 $\mathrm{kg\,mm^2}$ - a 86% decrease from the original (18.95 $\mathrm{kg\,mm^2}$) mostly due to the removal of metals and extensive thinning of material. The total decrease in mass of the parts shown in Figure 2.6 (excluding stepper motor) is 82% from 0.06kg to 0.01kg.

Improving Air Flow

In switching to the 3D printed ball structure, air-flow holes were able to be placed arbitrarily on the surface as long as the structural integrity was not compromised. You can see the chosen air-flow hole pattern in Figure 2.7. These holes, since they're printed directly from solid models, could be better controlled in position and size. The holes were increased from a 3mm diameter to 4mm diameter and covered a much denser pattern. The total air flow area in the ball surface increased 4.5x from 38$\mathrm{mm^2}$ to 177$\mathrm{mm^2}$, despite the overall ball



(a) Exposed surface of original ball with 3mm holes and 6% porosity

(b) Exposed surface of new ball with 4mm holes and 38% porosity

Figure 2.7: Comparison of original and new budger ball design, showing porosity and internal air-flow volume.

protruding surface decreasing by 31% (206mm^2) due to a slight reduction in ball diameter and exposed surface diameter. This resulted in a nearly 7x increase in porosity of 38.1% compared to the 5.7% of the original ball.

The increased ball porosity is also key in mitigating one issue inherent to the budger, the vacuum force that pulls air *around* the ball at the interface between the ball and the cap (see Figure 2.8). With sufficient gap, this can suck the edge of the fabric down into the budger or create pinching that dramatically increases the friction between the table and the fabric. As designed, the budger leaves only a 0.25mm radial gap, which results in 18mm^2 of surrounding flow area, or 9% of the total flow area. However, in realistic assembly the tolerance stack-up can be much worse. An added 1mm of vertical offset results in a 1.11mm gap and 18mm^2 of annular flow area (31% of total). Without the significant increase in ball porosity, the gapping could easily account for the majority of the air flow.

In addition, whereas the inside of the original ball was tightly packed with the stepper motor (see again Figure 2.6a), the structural changes made to reduce the rotational moment of inertial by turning the ball into a shell performed double duty by hollowing out the internals of the ball, providing a great deal more space for air to flow unimpeded and

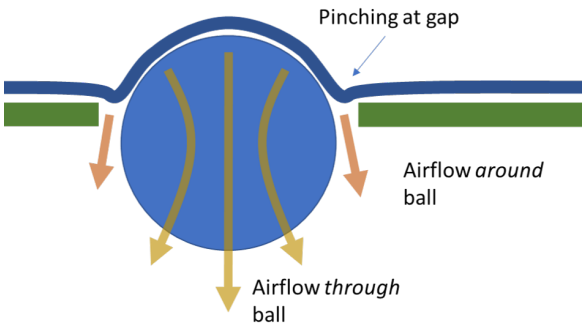


Figure 2.8: The gap between the ball and the budger cap allows air to pull around the ball as well as through it, which can pinch fabric increasing friction, impeding motion, or even getting fabric stuck with softer fabric types.

reducing pressure losses through the ball.

Surface Friction

The original ball was chosen to be silicon for its low durometer (squishiness), which made it difficult to machine, but provided good frictional interface with the fabric. Meanwhile, the new 3D printed parts, made out out nylon plastic, by their nature are very stiff. While the additive manufacturing of the ball significantly improved the structure, the one drawback is the stiffness of the printing material (nylon in this case, but "plastics" more generally) provides a fairly low coefficient of friction against fabrics. This was remedied by "painting" on a layer of air-cured silicon to the outer surface of the ball. This silicon was able to be completely customized for the appropriate durometer, uncured viscosity, cure time, and set time to make coating the balls more workable. With the added coating with a silicon layer with at 10 durometer rating, a painted surface was tested to have a coefficient of friction of 1 against denim (the target test material). After 2 years of testing a working table, the balls appear to have maintained their friction characteristics with minimal wear and tear.

To test the resulting frictional strength a small pulley apparatus was built to hang weight over the edge of the table and determine the amount of force it took for the fabric to slip. The weight was added gradually with sand into a lightweight cup (11g). This was tested using denim under various conditions, using 5 different sizes of fabric, each covering a

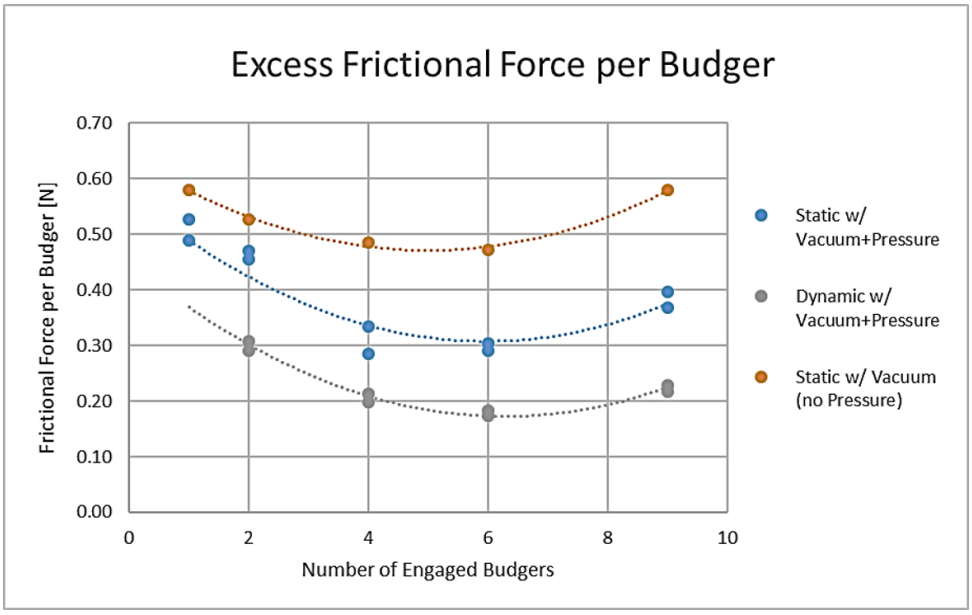
different number of budgers. The purpose of this test was not to fully characterize all possible fabric-budger-vacuum combinations, but to provide a general understanding of the available locomotive force provided by the budgers. By the design of the test, inducing slipping, the measured values represent the amount of force available in *excess* of what is necessary to move or hold the fabric.

Figure 2.9 has a summary of all the testing. A total of 5 fabric sizes were used, with each size covering roughly 1 budger per $0.044\mathrm{m}^2$ of fabric. For all tests, the vacuum through the budgers was on, and the positive pressure (discussed in Section 2.3.4) was on with the exception of one set of static tests. The tests were somewhat subjective in nature with criteria for slipping. In the static case, the budgers were covered with the piece of fabric and sand was added until there was consistent perceivable motion of the fabric, as the slipping over the budgers is more of a creep than a break. With dynamic testing the failure was harder to discern, but the set of budgers were driven forward and some slippage was allowed with the test stopping when the fabric could no longer reliably drive forward. Again, these tests are meant to simply provide a general qualitative understanding of the table's capabilities and not a strict characterization.

Drawing on the comments in Section 2.3.3, the excess suction space *around* the ball rather than *through* the ball should result in higher forces required to slip while stationary than when in motion since a portion of the force from the ball during motion is being used to overcome the suction around the ball. The data bears this out, with the dynamic force per budger ranging from 0.17 and $0.31\mathrm{N}$ ($0.23\mathrm{N}$ average) and the static (with pressure) ranging between 0.29 and $0.53\mathrm{N}$ ($0.39\mathrm{N}$ average). Additionally, as expected, by removing the positive pressure, there is both more friction of the fabric against the table *and* no lift to reduce the sealing suction around the ball, as a result the friction force increases to the range of 0.47 to $0.58\mathrm{N}$. That is to say that the friction forces generally behave as expected and the budgers provide sufficient locomotive force (in excess of what is required) to successfully manipulate the fabric.

Fabric Size	Fabric Weight	Number of Budgers	Fabric Area Per Budger	Fabric Weight Per Budger
m^2	g	#	m^2	g
0.0426	17.8	1	0.0426	17.8
0.0918	38.8	2	0.0459	19.4
0.1976	82.7	4	0.0494	20.7
0.2577	107.9	6	0.0430	18.0
0.3639	153.3	9	0.0404	17.0

(a)



(b)

Figure 2.9: Frictional force under varying conditions between denim and the budger table. Note, that as the size of the fabric increases so does the number of engaged budgers.

2.3.4 Reducing Surface Friction on Table

While table modifications aren't directly design changes to the budger, the introduction of the air table has implications for the budger and as such is discussed briefly here.

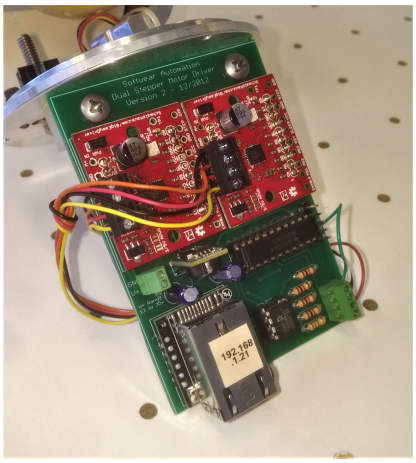
When trying to keep fabric (a very soft, easily deformable material) planar when in motion, the friction between the table and the fabric is the primary source of wrinkling. If the frictional force across a section of fabric is higher than its planar stiffness the budgers are no longer able to move the fabric forward. This friction and planar fabric stiffness is what sets the inter-budger spacing as well as governs the number of budgers needed in contact with the fabric. As is the practice in many industrial sewing shops, to significantly reduce this friction, the work surface was modified to become an air table.

Using the same testing as in the previous section, the friction force was measured between the fabric and the table with and without the air table being on as well as exclusively against the table and over the budgers. In testing, without air, the force to generate slipping was $2.14\mathrm{N}/\mathrm{m}^2$ with just the table laminate surface and $3.04\mathrm{N}/\mathrm{m}^2$ when the same fabric was placed over a set of budgers. Adding the air reduced the friction so significantly that it is essentially negligible, requiring much less than $12\mathrm{g}$ to slip ($12\mathrm{g}$ is the weight of the cup used to hold sand) or equivalently, less than $0.46\mathrm{N}/\mathrm{m}^2$.

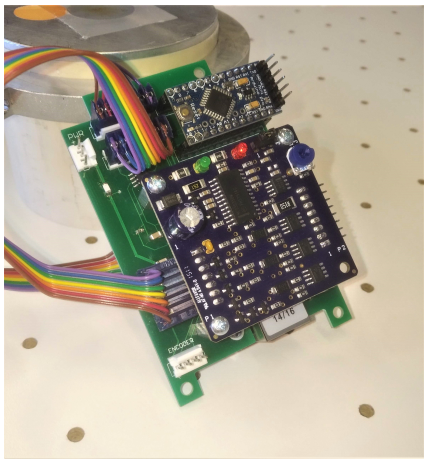
2.3.5 Updating the Electronics Package

The mechanical changes to the motors necessitated a slightly more complex electronics package. The entire package was redesigned from the ground up with low-cost components (as low as was feasible) and a custom printed circuit board (PCB). A comparison between the original and current electronics is shown in Figure 2.10.

The brushless motors require different drivers (from the previous stepper drivers). This change also required the addition of an encoder for position feedback on the budger orientation and local control computation on the microcontroller. To continue the theme of serviceability, the motor controllers and encoder interface use Molex style push-on con-



(a) Original electronics with stepper motor drivers (red PCB)



(b) New electronics with brushless motor drivers (blue PCB)

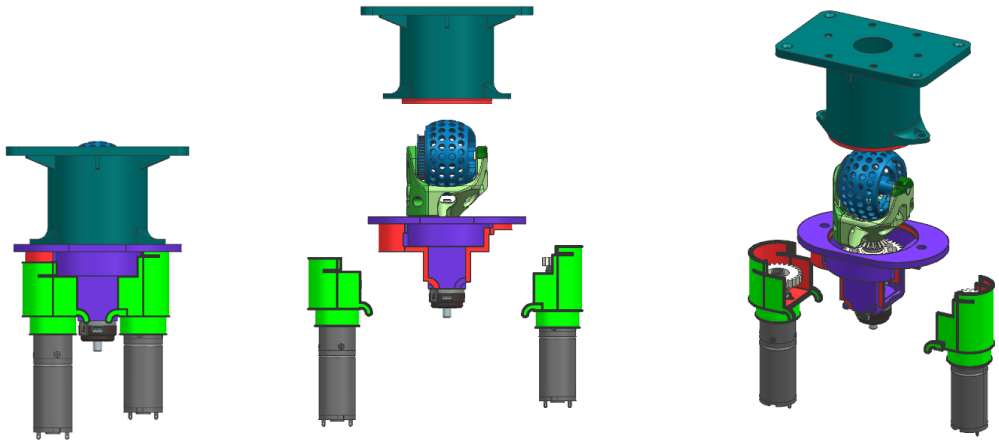
Figure 2.10: Comparison of original and new budger electronics

nectors to quickly separate the budger mechanics from the electronics for removal from the system. Similarly, the microcontroller (an Arduino Pro Mini clone) was itself removable should rapid replacement of firmware or replacement of a malfunctioning chip be necessary.

A continuously rotating budger presents both a challenge and opportunity in that the lack of a physical range limit means the budger could set its home or zero orientation in any direction relative to its housing. In addition to adding complexity to the microcontroller for control calculations, additional functions were enabled for the two way communication between the budger table and individual budger controllers as well as the ability to react to commands, such as re-zeroing the budger orientation (setting the current position of the budger as its zero position) in response to a command from the budger table. All of these commands are sent over a standardized JSON string to encode the speed, orientation, and any additional command/message which is decoded and executed by the budger.

2.3.6 Vacuum Sealing for Removable Parts

A core feature of the budger is pulling vacuum through the body to create a force between the fabric and the budger ball. When including quickly removable parts for easy servicing,



(a) (b) (c)

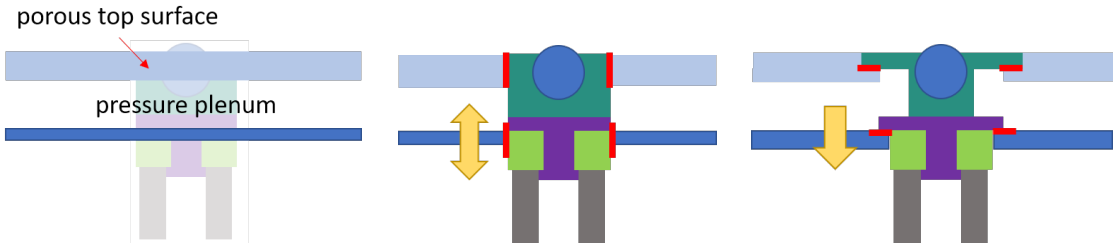
Figure 2.11: Sealing surfaces for removable parts

this presents a challenge, as every removable part presents an opportunity for gaps and loss of pressure. With that in mind, the new design minimized, as much as possible, the number of removable surfaces and size of sealing surfaces. Figure 2.11 shows the sealed budger along with two exploded views with each removable part disconnected and the sealing surfaces in red. Each sealing surface on the base (shown in purple) has a silicone lining. As the parts are connected, the corresponding surfaces provide a radial seal around each opening.

2.3.7 Fast Budger Removal with Integration to Air Table

Serviceability Motivation

The air table works by gapping the table surface and generating pressure plenum within this hollow section. This means that for a budger to be inserted into the table, rather than simply be held in place, it now has to bridge the two surfaces and seal against them to maintain the internal pressure. Along with the added design requirement, it is important the budger is still easily serviceable (i.e. quickly removable and replaceable from the table). Even at a very low failure rate, as the budger count grows, the likelihood of *some* budger *somewhere* on the table to fail also grows. For illustration purposes, assuming a 0.001% chance of



(a) Table cross-section with silhouette of budger for reference

(b) Radial sealing requires sealing both surfaces along with a means for holding the budger in place as it can slide vertically in both directions.

(c) Vertical sealing locks the budger in place with gravity and seals upon insertion.

Figure 2.12: Air-table sealing and insertion options with sealing surfaces shown in red.

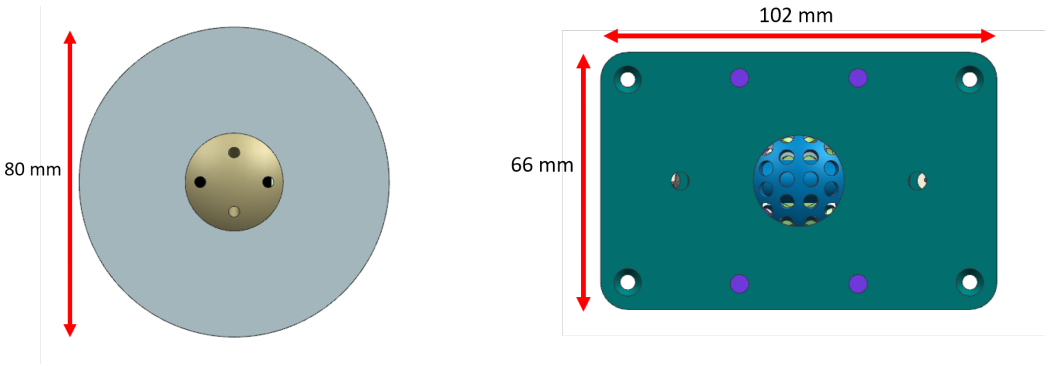
failure per day for any individual budger - at 100 budgers, there is a 10% chance of failure, and at 1000, a 63% chance a budger will fail somewhere in the factory any given day.

While the nature of the distributed system is inherently robust to individual failures, with mitigations like having fabric avoid a failed budger, it is desired to maintain a high system-wide uptime in any industrial application. For this purpose, the budger housing is designed to allow for a rapid remove-replace/repair cycle.

The primary challenges to keeping this cycle time low is the ease of access and complexity of disconnection. By design, the budger is fairly modular, with the mechanics and electronics completely separable. Quick disconnect connections were added for the motors, encoder, and vacuum tube. What remains is the physical challenge of removing the budger housing from the table.

Design Modifications

For affixing the budger to the table, the original design used a circular clamp beneath the table, which worked well for the cylindrical shape of that housing. As shown in Figure 2.12, there is a progression of possibilities for inserting and sealing the surface. Aside from the undesirable side effect of budgers falling out (see Section 2.3.2) if the clamp loses grip the air table complicates matters by extending the depth of the table (Figure 2.12a) and



(a) 82mm diameter cylinder held radially by clamp (adding an additional 20mm)

(b) 102mm x 66mm rectangular plate for drop-in design

Figure 2.13: Comparison of original and new table interface for the budger

additionally requiring an air-tight seal on *both* top and bottom surfaces of the table which the old budger would have to seal radially (Figure 2.12b). To remedy this, a pyramid-like or tiered structure was chosen such that each surface can seal against 1 surface of the table, with the smaller sealing surface able to pass through the larger hole in the table. This configuration, shown in Figure 2.12c, allows for sealing vertically against both surfaces simultaneously, which mean sealing is accomplished just by the act of inserting the budger.

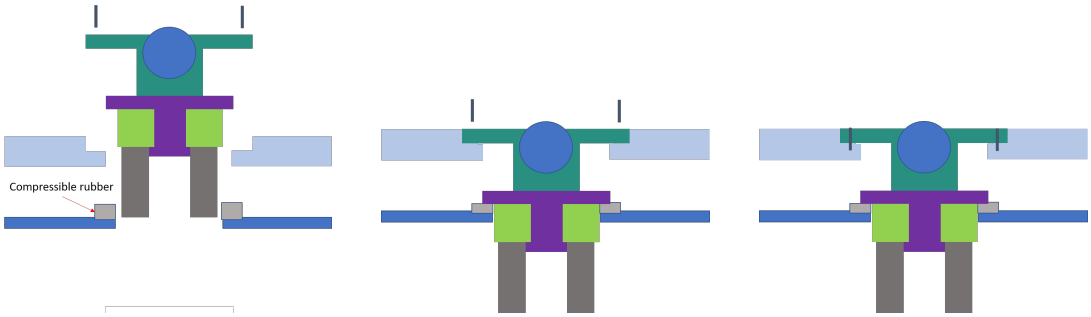
The budger housing assembly needed to be adapted to this purpose. Continuing with the desire for easy access and given the opportunity to use gravity as a benefit, the budger was redesigned to drop-in from above. The budger housing was modified (shown in difference between Figures 2.12b and 2.12c) by adding flanges to the housing (purple) and cap (green) to accommodate the required tiered structure: the body added a small lip around it to catch the inner surface while the cap added a rectangular plate extension. Because this extension takes up space on the top (active) surface of the air-table, a ring of holes were added around the plate for air flow to continue providing upward pressure near the budger ball.

These changes impact one of the key design requirements to keep the footprint low such that budgers have the ability to be placed close together (if needed). Figure 2.13 compares the top-down dimensions of the budger designs and shows the implications of drop-in. Due

to the out-boarded motors extending radially from the center axis, one dimension of the new budger design had to be longer than the other with the new budger taking up 6732 mm^2, while the old budger only takes up 5281 mm^2. However, this is a somewhat unfair comparison as the space around the cylinder of the original budger is mostly unusable. When accounting for a rectangular budger array, both budgers take up almost the same exact space (6732 mm^2 and 6724 mm^2 for new and original respectively). Further, this doesn't account for the additional radial space of about 10mm required for a clamp (and the fact that a wall of cylinders packed directly next to each other would slice the table apart), making the average table space for the original budger roughly 10,000 mm^2. As a result the general placement density is maintained or improved upon with original budgers requiring about 100mm between neighboring budgers, while the new design can be as close as 70mm along the minimal dimension and just over 100mm along the larger.

One challenge to this approach is the uncertainty in table depth and tolerance on the budger housing assemblies - specifically the distance between the two sealing surfaces. The budger top also needs to remain flush with the table to keep fabric from catching while in transit. Figure 2.14 shows the final makeup of the budger/table interface with the pocketed upper surface and a rubber seal along the bottom. The soft rubber provides cushion to take up any variability in depth by compressing as the budger is inserted and simultaneously providing a good seal for the bottom surface. To provide compression force and keep the budger firmly in place screws holes were added at the corners of the upper plate.

The final removal procedure involves disconnecting the budger from the electronics board, removing the screws, and pulling the budger out vertically from the table. As the budger is flush with the table, the holes providing air flow can be used as an access point for tooling to grip the budger housing. Since the electronics stay in place, a tested and fully functional budger can be dropped in place of the failed budger or, because of the quick disconnect capability of the motors (Section 2.3.1), a failed motor pod can be exchanged. This full process takes under 1 minute and can possibly be performed even without stopping



(a) Budger inserted from above with rubber sealing surface inside the table

(b) As budger is inserted, the rubber compresses to seal and fill in gaps due to tolerance stack-up

(c) Screws inserted from top to seal top and provide compression force on rubber

Figure 2.14: Budger-table assembly

the machine.

2.3.8 Cost Analysis for Industry Feasibility

One design parameter for this system was to be viable for use in industry. Part of that requires keeping the costs competitive with existing automation options.

The table in Figure 2.15 provides a full cost breakdown of the budger, as-built, including the electronics. In low quantities for the single prototype budger table, the full budger costs about $400, putting a full table of 30 budgers around $12,000. While this price might be reasonable given the capability and workspace the budger table provides, this doesn't include any price breaks for a larger scale manufacturing operation. Scale pricing was also examined, using either quoted or estimated breaks for large orders (greater than 1000 parts) or assumptions on costs for manufacturing changes (eg. injection molding for the housing). Figure 2.15 also shows the estimated cost with these estimated reductions, bringing the budger down to under $100, with the majority of the gains due to the motor, housing, and electronics (see Figure 2.16). At this price, a full budger table comes closer to $4,000. This puts it very competitive in price with a simple 3 axis gantry. For reference, even "low cost" belted drives run about $1/mm. At that rate, the x-y axes of a gantry, covering the same workspace as the prototype table ($2\mathrm{m}$ x $1\mathrm{m}$)and using dual rails for the long axis, costs

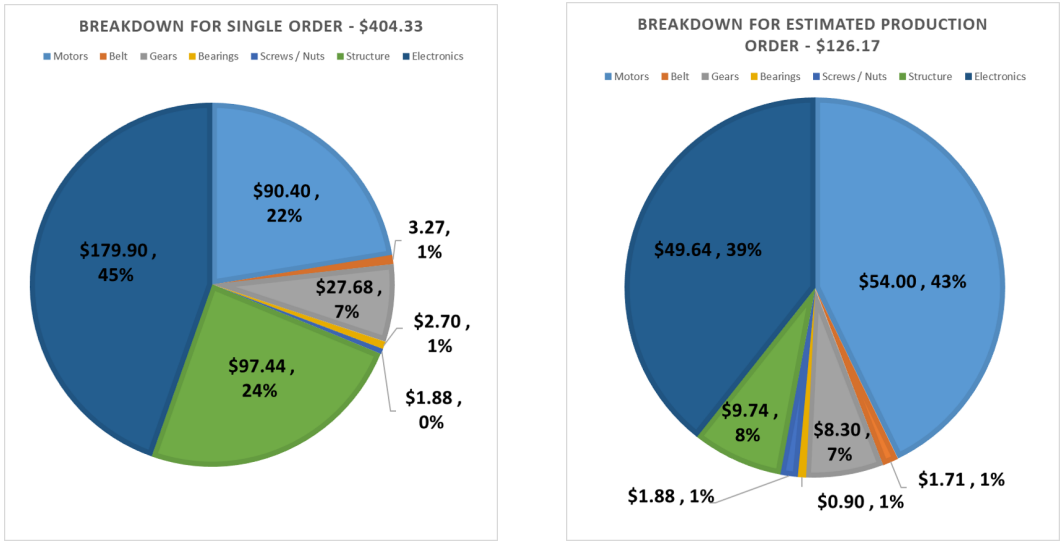
	Description	QTY	Price 1	Price > 100	Lot	Single	>100pc	Single	>100pc	Estimated Price at Scale Break	Estimated Price at Scale Price
Motors	brushless motor	2	$ 45.20	$ 30.00	1	$ 90.40	$ 60.00				
							Subtotal	*$ 90.40*	*$ 60.00*	*10%*	*$ 54.00*
Belt	belt to drive ball	1	$ 3.27	$ 1.71	1	$ 3.27	$ 1.71				
							Subtotal	*$ 3.27*	*$ 1.71*	*0%*	*$ 1.71*
Gears	miter-spur	1	$ 2.80	$ 2.80	1	$ 2.80	$ 2.80				
	miter-belt	1	$ 2.34	$ 2.34	1	$ 2.34	$ 2.34				
	shaft spur	1	$ 2.60	$ 2.60	1	$ 2.60	$ 2.60				
	turn motor spur	1	$ 3.07	$ 3.07	1	$ 3.07	$ 3.07				
	spin motor spur	1	$ 3.87	$ 3.87	1	$ 3.87	$ 3.87				
	shaft	1	$ 3.00	$ 3.00	1	$ 3.00	$ 3.00				
	round stock	1	$ 10.00	$ 10.00	1	$ 10.00	$ 10.00				
							Subtotal	*$ 27.68*	*$ 27.68*	*70%*	*$ 8.30*
Bearings	plastic bushing - sleeve	2	$ 0.49	$ 0.15	1	$ 0.98	$ 0.30				
	plastic bushing - flange	4	$ 0.43	$ 0.15	1	$ 1.72	$ 0.60				
							Subtotal	*$ 2.70*	*$ 0.90*	*0%*	*$ 0.90*
Screws / Nuts	M4x8 (cap to base)	2	$ 6.44	$ 6.44	100	$ 0.13	$ 0.13				
	M2x6 (affix motor to motor m	4	$ 6.80	$ 6.80	100	$ 0.27	$ 0.27				
	M2.5x8 (affix everything else)	12	$ 4.80	$ 4.80	100	$ 0.58	$ 0.58				
	M4 nut	2	$ 10.00	$ 10.00	200	$ 0.10	$ 0.10				
	M2.5 nut	4	$ 10.00	$ 10.00	50	$ 0.80	$ 0.80				
							Subtotal	*$ 1.88*	*$ 1.88*	*0%*	*$ 1.88*
Structure	base	1	$ 28.99	$ 28.99	1	$ 28.99	$ 28.99				
	cap	1	$ 19.85	$ 19.85	1	$ 19.85	$ 19.85				
	motor mount	1	$ 19.16	$ 19.16	1	$ 19.16	$ 19.16				
	ball	1	$ 13.90	$ 13.90	1	$ 13.90	$ 13.90				
	spindle (part A)	1	$ 7.94	$ 7.94	1	$ 7.94	$ 7.94				
	spindle (clamp)	1	$ 1.84	$ 1.84	1	$ 1.84	$ 1.84				
	spindle (part B)	1	$ 5.76	$ 5.76	1	$ 5.76	$ 5.76				
							Subtotal	*$ 97.44*	*$ 97.44*	*90%*	*$ 9.74*
Electronics	grab bag of components (est.)	1	$ 15.00	$ 15.00	1	$ 15.00	$ 15.00				
	standoffs and such	1	$ 3.00	$ 3.00	1	$ 3.00	$ 3.00				
	printing of circuit board	1	$ 30.00	$ 30.00	1	$ 30.00	$ 30.00				
	Adruino Pro Mini	1	$ 4.95	$ 4.95	1	$ 4.95	$ 4.95				
	Encoder interface (counter)	1	$ 5.00	$ 3.00	1	$ 5.00	$ 3.00				
	brushless driver	2	$ 41.00	$ 41.00	1	$ 82.00	$ 82.00				
	Encoder	1	$ 32.00	$ 19.56	1	$ 32.00	$ 19.56				
	Ethernet module	1	$ 7.95	$ 7.95	1	$ 7.95	$ 7.95				
							Subtotal	*$ 179.90*	*$ 165.46*	*70%*	*$ 49.64*
							Total	*$ 403.27*	*$ 355.07*		*$ 126.17*

Figure 2.15: Full cost analysis of budger design with part breakouts and estimates for large scale manufacturing prices.

$5,000 without accounting for z-axis and electronics, while still having all the drawbacks of the gantry mentioned in Section 2.1.1.

2.4 Conclusion

Although considered an "update", the changes made to the budger between the original version and the updated version amount to a complete redesign of the budger from the ground up with the new design retaining no components of the original beyond the concept of a steerable ball through which vacuum is pulled. Every design comes with tradeoffs where not every goal can be met, but as shown in Figure 2.17 the changes made during the redesign resulted in significant improvements across the board. Returning to the goals



(a) (b)

Figure 2.16: Comparison of cost makeup breakdown by value and percentage for both low quantity "single-order" for prototyping and estimated large-scale production costs.

set in Section 2.1.4, the new design hits every target. Although not quantified, the general capability of the budger through improved range of motion allows for arbitrary and continuous motion of the fabric while simultaneously translating and rotation. Serviceablility was addressed through both the easy exchange of motors by keeping them outboard of the central housing as well as making removing and replacing budgers easier and more accessible through the drop in design. With an average reduction of about 80% for both the spinning and turning mass and moment of inertias, coupled with more efficient motor choices, the ball spin speed improved almost 8x to just over 0.5m/s, while the point to point turn rate from any direction to any other was reduced to 250ms. Changing the motors and using onboard microcontroller for local closed-loop control allowed the budgers to hold position with full torque when needed. As a follow on benefit, this design means that torque is only used as needed and as a result, power consumption was reduced by over 80%. While the one area that didn't improve was exposed footprint (the area of the budger consuming space on the top table surface), the general required footprint for assembly decreased enough to allow a 50% higher budger placement density if required.

Measurable Performance Characteristic	Units	Original	Current	Raw Change	Percent Change	desired
ball spin moment of inertia	kg-mm^2	18.95	2.67	-16.28	***-85.9%***	-1
ball spinning mass	kg	0.06	0.01	-0.05	***-82.4%***	-1
spindle rotation moment of inertia	kg-mm^2	54.71	11.23	-43.48	***-79.5%***	-1
spindle rotating mass	kg	0.17	0.05	-0.12	***-68.7%***	-1
exposed ball air-flow area	mm^2	38.35	177.09	138.73	***361.7%***	1
exposed ball porosity	%	5.7%	38.1%	0.32	***566.3%***	1
table surface area per budger	mm^2	6221	6732	511.00	***8.2%***	-1
maximum budger density	budgers/m^2	96.88	148.54	51.66	***53.3%***	1
average power consumption	Watts	38.40	6.72	-31.68	***-82.5%***	-1
ball spin speed	m/s	0.06	0.52	0.46	***766.7%***	1
spindle turn rate	s	1	0.25	-0.75	***-75.0%***	-1

Figure 2.17: Summary table of changes in performance characteristics from original to redesigned budger.

CHAPTER 3

REALTIME VISUAL FEEDBACK FOR FLEXIBLE MATERIAL

3.1 Motivation

Feedback Needs Because each of the budgers are fully independent actuators, they can be locally activated to manipulate the fabric. An optimal system should handle many pieces of fabric at once such that each piece of fabric can utilize as few actuators as needed to perform the desired manipulation task, freeing up the rest to handle other fabric tasks. To handle this type of control, the feedback must include the existing boundary of the controlled fabric (full boundary, even if portions are occluded) to determine what subset of budgers to activate.

In order to provide such a boundary, the feedback system requires the fabric flat-pattern (ideal shape) to either be provided via communication from an upstream manufacturing process or by matching to a pattern database from the initial fabric observation. Because this information is used for closed-loop control, the matching/tracking operation must execute as quickly as possible to minimize any delay or prediction errors. Due to tolerances in the fabric cutting process, the detection and tracking algorithm should be robust to small scaling and shape errors.

Wrinkle tracking The primary difficulty in automating the garment manufacturing industry is the complexity in machinery handling flexible material – a category which most (comfortable) fabrics would fall under. Computing deformable body dynamics, unfolded shapes, and ideal grasping locations to maintain rigidity are all complex and computation intensive processes. However, when limiting the work regime to certain sewing operations, which are well-defined and take place on a table (flat surface), the problem can be translated into maintenance of a fabric in a known "rigid" shape. Indeed, as humans sew a garment,

they maintain local tension to keep the fabric flat as it passes under the needle, which is important to perform at all times, to avoid bunching under the needle.

On a sewing table, deformation of a fabric must be accompanied by wrinkles or ridges as the fabric is pushed up out of the plane of the table. Maintaining shape rigidity of such a deformable body, can be translated to the equivalent problem of ensuring there are no wrinkles in a manipulated fabric (or quickly removing them should they form).

In previous work, Ryder Winck established a prototype distributed actuator based system to manipulate fabric (in conjunction with a comprehensive sewing system with automated feeder dogs) [11]. While this initial work used open-loop control to impart translation and rotation to fabric, this thesis extends it by implementing upgraded actuators (Chapter 2) and a vision feedback system for closed-loop control. A schematic of this system is provided in Figure 1.

While the dynamics of distributed actuation systems have been studied previously [12][14], these systems use force balance on the conveyed body under the assumption of slipping friction by counter-rotating some actuators. Naturally, in the case of a deformable body, this is impossible and would immediately deform the body. As such, rigid-body kinematics are used to compute the actuator velocities. This caveat of the assumption of the maintenance of a rigid body, required for the control described in Section 4.4 is somewhat implausible under general real-world conditions. By relying solely on rigid-body kinematics, any errors in the actual speed/positioning of the actuators, unexpected friction on the table, or any number of other catches or unpredicted behavior have the potential, and indeed a high probability, to generate deformation in the fabric, resulting in wrinkles. Thus, the wrinkle detection and elimination (Sections 3.4 and 4.5) is essential for the proper function of robust fabric control by distributed actuation.

The constraints imposed on the design of the system are the same as those mentioned above, namely, that this system is intended for rapid implementation in manufacturing in which low cost and high speed are of high importance. To achieve this, readily available,

off-the-shelf components were chosen: a Kinect for Xbox One (for depth and color imaging) operating with a moderately powered Windows 10 computer.

3.2 Related Work

Template Matching While there is a great deal of work related to object recognition and tracking using complex neural networks, this project aimed for a simple algorithmic approach to provide a known outcome. As basic image processing capabilities, like segmenting an image and extracting edges, are readily available in various software packages, the primary focus was on matching a partially occluded polygon to its ideal base polygon.

Much of the available literature on recognizing and matching polygons in a scene are relatively old computer vision papers. There have been various disparate methods for these purposes. In one case occluded shapes are compared against successively generated hypothesis shapes [17]. Other papers propose recognizing shapes based on their contour. These methods are often for more complex shapes (not low number vertex polygons), which may use direct equation solution by comparing the visible contour moments [18] [19], or use inflection points to segment the shape and identify strings of segments to define a unique polygon [20]. Another method uses a graph based approach to identify a closed-form solution to the best match of points that lie on the edges of a polygon [21]. Other matching algorithms reformulate the problem by converting it to a straightness measure to simplify the computation - but can not handle occlusion [22].

Each of these are not particularly well suited for real-time feedback as the algorithms are either extremely complex, computationally expensive, unable to handle occlusion, or some combination thereof.

Another method of object recognition uses 2D image segmentation in conjunction with existing 3D object models to do pose estimation [23], but still suffers from complexity and speed.

Looking to the 3D realm for inspiration, though, provides the iterative closest point

(ICP) algorithm [24] [25] which can quickly find local minimum by matching sampled points to a known surface/model. David Simon provides a good overview of ICP and its practical applications in his thesis [26]. While not initially developed for 2D models, it is possible to use ICP to align polygons in which each ”point” is one of the vertices of the polygon. Additionally, several papers have examined improvements to the ICP algorithm. such as sparse ICP by choosing subset of points to reduce errors in noisy or incomplete data [27], however the issue in polygon matching is the *lack* of data points. Another paper providing inspiration introduces efficiency improvements with different methods of registration between points [28], which leads toward the method settled on for this paper.

Wrinkle Detection To this researcher's knowledge, there are no other distributed manipulation systems operating on fabrics. However, there are a number of related studies in the field of garment deformation or detection of wrinkles.

A large amount of work, uses a depth sensor to map an observed folded or draped garment and applies various processing techniques to then unfold the garment. This may be by image segmentation to order unfold maneuvers [8], comparing against known 3D shell models [6], or determining optimal re-grasp points [7]. Each of these scenarios require an expensive and complex robot arm (or set of arms) to perform grasping maneuvers which can be slow and laborious, which would not be useful for real-time “in-transit” wrinkle correction.

Another paper [4] looked at the case of identifying wrinkles, or more specifically creases, in a marginally flat fabric for the purpose of ironing. The image processing focused mostly on discerning the difference between general non-flatness and creases using machine learning. This again is relatively slow and operates on stationary fabric (which uses a resolution higher than needed for the general transportation case).

3.3 Fabric Tracking via ICE2 Algorithm

3.3.1 Flat-Pattern Tracking Simplifications and Assumptions

Sew Process Advantages Because sewing is an inherently two dimensional process (most garment manufacturing operates on a flat sewing surface) the "object tracking" problem is reduced to a polygon tracking problem, as any article of fabric is defined by a 2D "sew pattern". The manufacturing space/sewing table is a partially controlled environment of known dimensions, so the overhead camera can apply perspective correction and be calibrated to convert from the digital image pixels to real value coordinates (in meters). That is, any information extracted from the camera can be assumed to be in its true shape (eg. extracted edges of a square piece of fabric result in undistorted square edges in real coordinates after passing through the conversion).

Assumptions of existing computer vision capabilities The sewing environment provides a number of possible physical solutions to the problem of identifying the fabric against the backdrop of the table or other pieces of fabric and therefore this is not discussed in depth. There are assumed to be a number of existing computer vision methods to segment the fabric from the table (which is perspective corrected) and to extract the boundary/describing polygon.

In this implementation, OpenCV, an open source computer vision library, is used for all the image processing and detection tasks. Specifically, fabric is identified within the image by its color and segmented with blob detection. The boundary is then extracted from the blob and converted to a convex hull (for robustness to interior projected occlusion like placing a hand on the fabric), which reports the vertices of the resulting "observed" shape.

This assumption allows the tracking operation to be effectively separated from the image processing operation, which is free to change with any improvements in computer vision algorithms, or to be tailored to specific use cases (which may change or could be

optimized in different manufacturing environments).

3.3.2 ICE2 Algorithm

Overview The general goal of the tracking algorithm is to overlay the expected or ideal template of the polygon defining the true fabric shape over the observable portion of fabric in the frame. To do this, the general overlay procedure works as follows:

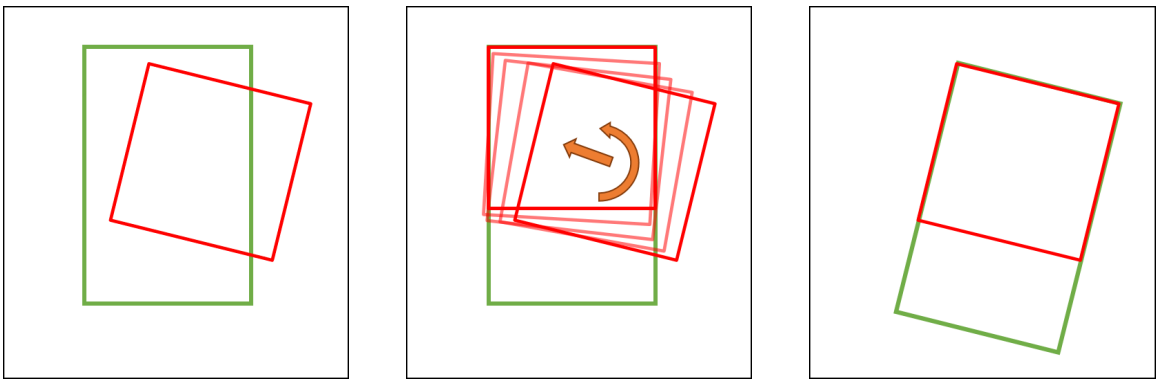
The system is either given or is assumed to have identified the template polygon defining the fabric shape which is used as the base "true" polygon stored as a collection of vertices positioned in 2D space in the current expected location of the fabric. In each iteration, the image-processing returns the vertices of a polygon that define the observed portion of fabric, which are positioned in the same 2D space. Then some algorithm spatially aligns the observed polygon with of the template polygon and returns the resulting rotation and translation that was applied to the observed polygon. Finally, the inverse of that rotation and translation is applied to the template, which is then the new expected position of the fabric and should overlay the original observed polygon.

This high-level procedure is outlined in algorithm 1 and depicted in figure 3.1 for clarity.

What follows in this is a description of the algorithm to generate the rotation and translation that define the alignment between the observed polygon and the template polygon.

Algorithm 1 Fabric Tracking Algorithm

1: **procedure** ICE2TRACK(P_m, P_t) ▷ align template vertices, P_t, to the observed vertices, P_m
2: (*optional*) $P_{t,pred} \leftarrow$ Predict forward expected P_t
3: $R, T \leftarrow$ ICE2($P_m, P_{t,pred}$) ▷ Retrieve rotation and translation from the ICP algorithm, replacing ICPalign (algorithm 3) with ICE2align (algorithm 4)
4: $P_{t,align} \leftarrow R^{-1} * P_{t,pred} + T^{-1}$ ▷ Apply inverse rotation and translation to the template vertices
5: (*optional*) $P_{t,corrected} \leftarrow$ Correct $P_{t,pred}$ with $P_{t,align}$
6: **return** $P_{t,corrected}$ ▷ Return aligned template vertices
7: **end procedure**



(a) Observed polygon in red, with expected location of template in green

(b) Align observed polygon to template with ICE2 algorithm

(c) Apply the inverse rotation and translation to the template

Figure 3.1: General overview of the entire fabric positioning algorithm once receiving the extracted fabric vertices

Polygon Alignment The alignment algorithm is based on the ICP algorithm [24][25] in which each sampled point is matched with the nearest target point (a brief summary of the implementation is provided as algorithm 2). For the 2D polygon case, the sampled points are the vertices of polygon, which would be matched with the vertices of the target polygon. In the fabric tracking application, these vertices are extracted from the convex hull around the observed fabric, as described above.

Algorithm 2 Iterative Closest Point

procedure ICP(P_m, P_t) ▷ match set of measured points, P_m to set of target points P_t
  initialize R, T, err ▷ rotation matrix, translation vector, error value
  while $err \geq tolerance$ **do**
    for $i = 1$ to $length(P_m)$ **do**
      $V_{align}[i] \leftarrow ICPalign(P_m[i], P_t)$ ▷ add vector to set of alignment vectors, V_{align}
    end for
    compute Singular Value Decomposition (SVD) of V_{align}
    extract T_{iter} and unscaled R_{iter} ▷ R_{iter}, T_{iter} are rotation and translation for this **while** iteration
    $err \leftarrow Norm(V_{align})$ ▷ compute current error
    $R \leftarrow R * (\delta * R_{iter})$ ▷ apply δ scaled rotation
    $T \leftarrow T + (\delta * Titer)$ ▷ apply δ scaled translation
  end while
  return R, T ▷ return the rotation and translation
end procedure

Algorithm 3 ICP alignment vector

1: **procedure** ICPALIGN(p_m, P_t)
2: $v_{align} \leftarrow vector$ from p_m to nearest p_t in P_t
3: **return** v_{align}
4: **end procedure**

The vertices of the observable region of an occluded polygon must lie on the edges of the target polygon (see figures 3.2 and 3.3) and may also result in additional vertices being generated. With such a small number of usable points, these vertices are no longer able to match up in a near 1-to-1 basis and the ICP algorithm will result in a poor match (an unnecessary offset/rotation) shown in figure 3.4. Rather than match vertices of the target polygon, the proposed algorithm matches an observed vertex to the nearest target edge, using an error distance normal to the edge, similar to the normal projection method described in [28].

This method allows the occluded polygon to position and align with the target polygon, but can result in a case where it reports zero error, when it's not optimally placed. For example, in the case of a rectangle, there will be no error in alignment when all vertices lie on the target edges, which allows for the occluded portion to be anywhere along the length of the rectangle (see figure 3.5).

Edge Locking To deal with such a case, the algorithm is further modified to introduce an affinity for "edge locking" in which observed edges have a preference to lie on the nearest template edge. To accomplish this, each observed vertex is matched perpendicularly to the two nearest edges on either side of the closest point. The two error vectors, $v_{err,n}$, are combined via a weighted average in which the weight, $w_{err,n}$, is determined by a Gaussian function (equation 3.1) with a tunable "capture radius", r_{cap}. This capture radius has the effect of determining the relative competitive strength that each template edge has to attract

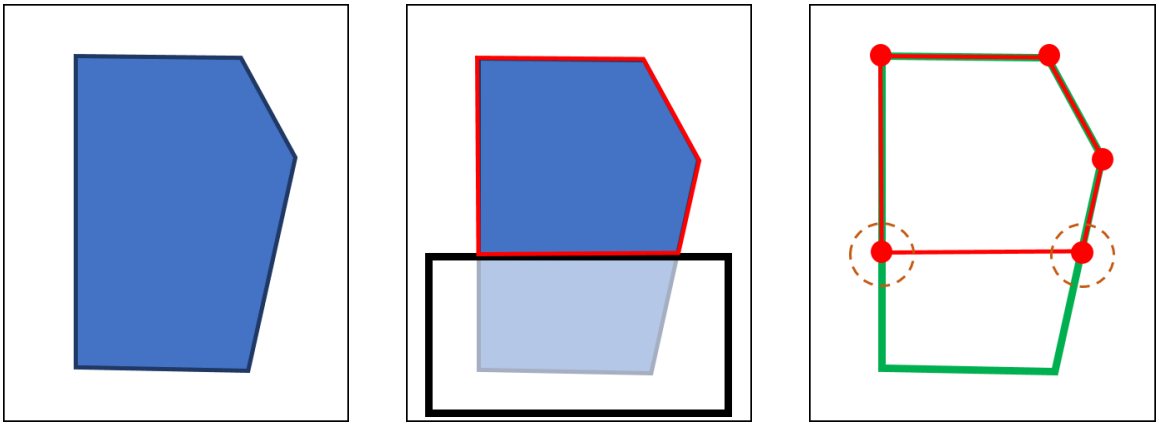


Figure 3.2: Example of occlusion resulting in extracted vertices lying on either the vertices of the template polygon OR its edges. (a) Initial fabric, (b) Fabric occluded at bottom (shown as white box with black border) with camera-visible portion outlined in red, (c) Extracted vertices from occlusion overlaid on template shape ("new" vertices lying on template edges circled in orange)

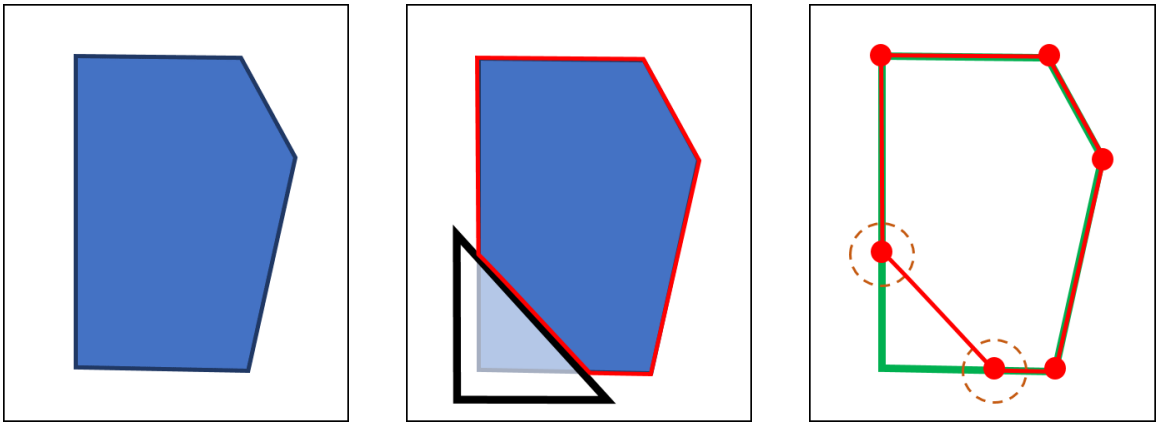


Figure 3.3: Example of occlusion resulting in additional vertices (6 observed, 5 expected). (a) Initial fabric, (b) Fabric occluded at bottom corner (shown as white triangle with black border) with camera-visible portion outlined in red, (c) Extracted vertices from occlusion overlaid on template shape ("new" vertices lying on template edges circled in orange)

an observed vertex.

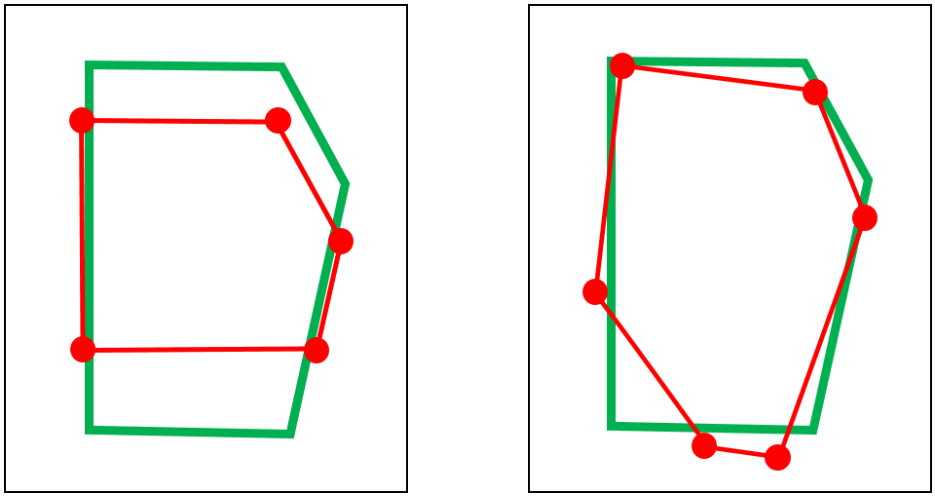
$$w_{err,n} = e^{-\frac{|v_{err,n}|^2}{2r_{cap}^2}} \tag{3.1}$$

Again taking the case of the occluded rectangle from figure 3.5 for example, figure 3.6 depicts how the algorithm progresses. This weighting puts more emphasis on the smaller error vector pushing the observed vertex quickly toward the closest edge as seen in figure 3.6b. Because of the observed vertex proximity to one of the edges, the weighted average error vector primarily points in that direction. The overall result is such that these weighted combinations have a tendency to force observed vertices into a corner (if possible).

This alteration adds one complication in the case that when the observed vertex lies external to an edge, such that the normal projection from the vertex to the second nearest edge does not intersect with the edge (as shown in figure 3.6c). In this case, the error vector is computed between the observed vertex and the nearest template vertex on that edge (which turns out to be the already computed closest vertex).

As a result of the edge locking additions, the algorithm first quickly converges to the case where each observed vertex lies on a template edge and then (as seen in figure 3.6e) the relative weights between the remaining vertices not yet locked into a corner (template vertex) continue to push the occluded polygon toward the edge, with the final "locked" position shown in figure 3.6f. This is where the tuning of the capture radius is most critical. If the radius is both too small or too large the error vectors will be similarly weighted and, in the case depicted, cancel each other out such that the iterative algorithm would terminate before reaching the minimum error position.

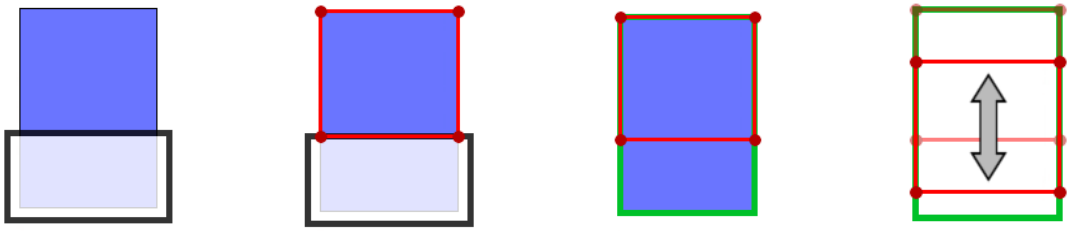
Tracking Further improving the efficiency of the algorithm, and inherent to the application, is the use of prediction in tracking these polygons as they move. This efficiency is primarily dependent on how closely the observation and the template are positioned initially. A closer initial placement reduces the total number of iterations needed to get to a



(a) ICP matching result for case in figure 3.2

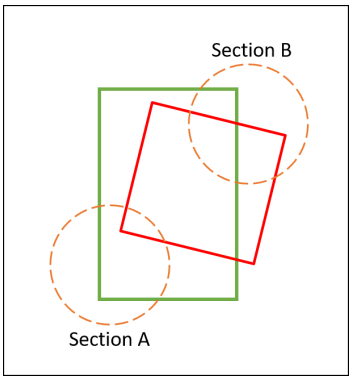
(b) ICP matching result for case in figure 3.3

Figure 3.4: Example of occlusion resulting in additional vertices (6 observed, 5 expected)

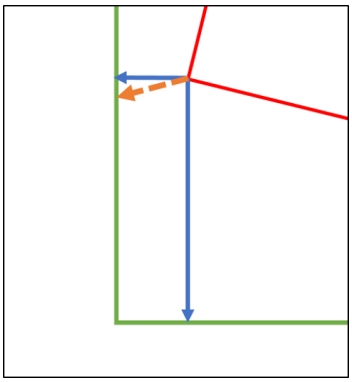


(a) (b) (c) (d)

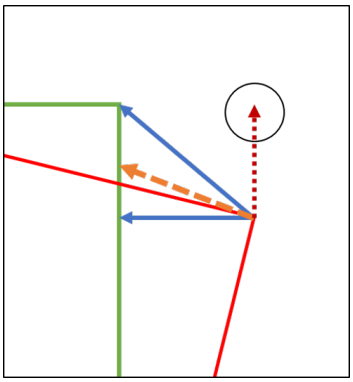
Figure 3.5: With simple shapes, using an error vector between the observed vertex and closest template edge can allow for multiple solutions: (a) Fabric (in blue) with occlusion shown at bottom (white region outlined in black), (b) Vertex detection of visible area (red), (c) Detection overlaid with ideal template (green), (d) Possible solutions with pure ICP to edge (all positions where vertices lie on the edges are valid)



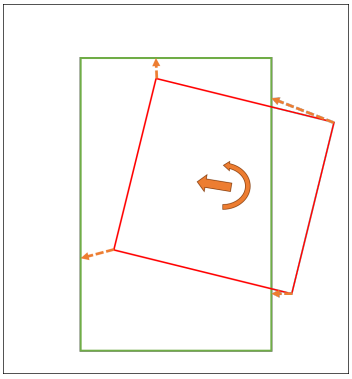
(a) Initial position of template (green) and observed vertices (red) with regions of interest circled in dashed orange



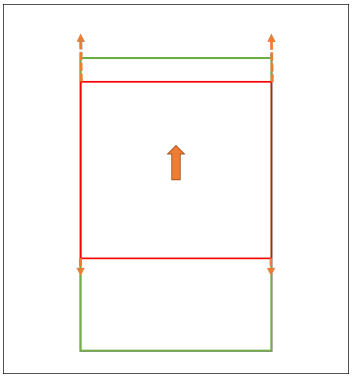
(b) Section A: Error vectors (blue) with weighted resultant error (orange dashed)



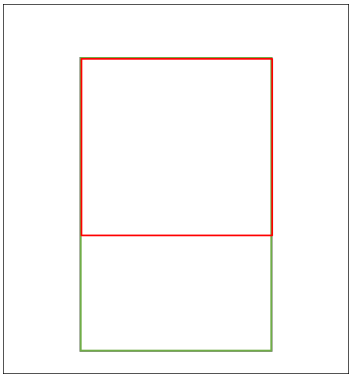
(c) Section B: Error vectors (blue) with weighted resultant error (orange dashed) showing failed normal error (red dotted)



(d) Full error vectors with resulting rotation and translation applied at center



(e) After alignment, remaining error vectors push the observed vertices toward the upper edge



(f) Final locked position

Figure 3.6: Overview of the introduced ICP variant using weighted point-to-edge error vectors

minimal error (within a tolerance) configuration. For any tracking operation, it is likely the tracking system will already employ some kind of filtering algorithm like a Kalman filter. The predict step of Kalman filter can then be used to apply an initial transformation to the template polygon reducing error between observed and expected position, in turn reducing the number of computations before convergence. A follow-on benefit is that the quicker the algorithm runs, the smaller the time-step between predictions, which further increases prediction accuracy.

Identification Lastly, this algorithm can also be used to identify the observed polygon from a database of possible polygon, under the assumption that the initial observation does not include any occlusions (eg. initial fabric detection takes place in a known obstruction free zone). Because the algorithm inherently calculates an error vector it is simple to construct a "match accuracy" score by taking some combination of error vectors (eg. sum of squares) with which to compare between different templates to determine the best match. However, as any iterative closest point variant will terminate at a local minimum, the identification algorithm applies some conditioning and redundancy.

First it's possible to cut down the set of database matches by filtering the templates to only those with the same number of vertices and within a size range tolerance (applied to internal area). Then for each possible match, the observed polygon is shifted such that its center of mass is coincident with that of the template polygon. Finally, to avoid local minimums, the alignment algorithm is applied from various initial starting orientations (rotational offsets) with the lowest "match accuracy" score and orientation being saved for each possible template. These changes naturally require additional computation time, but since they are done only once upon for initial identification, the delay is negligible (still completing on the order of 0.10s).

3.3.3 Performance

Algorithm 4 ICE2 vector

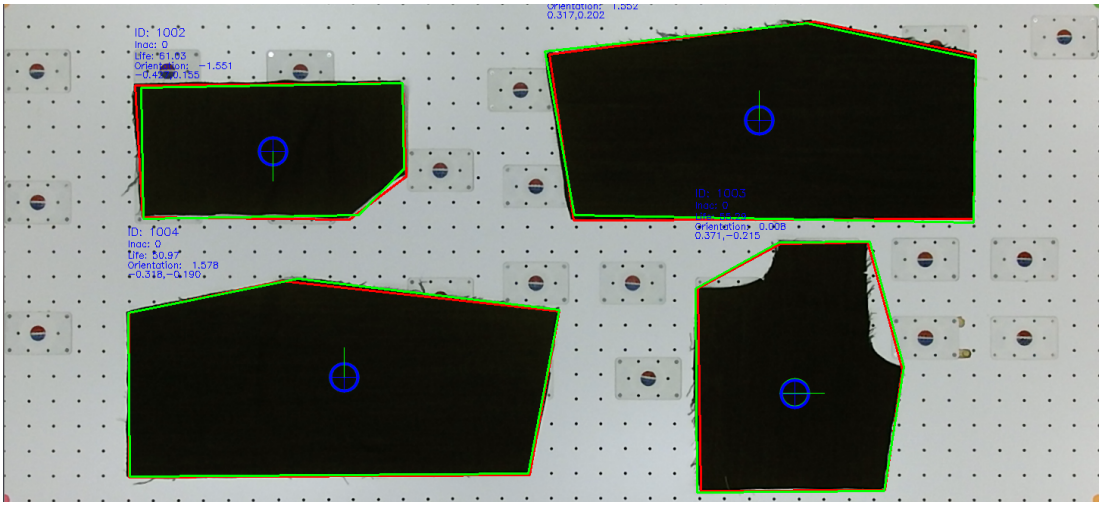
```
1: procedure ICE2VECTOR(p_m, P_t)   ▷ generate the 2-edge weighted vector from point p_m to
                                        the nearest 2 edges connecting the set of points, P_t
2:     p_t ← point in P_t nearest to p_m                ▷ Obtain nearest template vertex
3:     E ← edges adjacent to p_t                        ▷ Retrieve the edges on either side
4:     for i = 1 to 2 do
5:         v_temp ← ⊥ vector from p_m to E[i] ▷ Generate vector from the point to the edge that is perpendicular to the edge
6:         if v_temp lies outside template polygon then
7:             V_edge[i] ← vector from p_m to p_t ▷ When the perpendicular vector intersects outside the edge, replace with vector to nearest template vertex (see figure 3.6c)
8:         else
9:             V_edge[i] ← v_temp                   ▷ Otherwise keep the perpendicular vector
10:        end if
11:    end for
12:    v ← weighted average of V_edge               ▷ Weight each vector by the equation 3.1
13:    return v                                     ▷ Return the weighted combined vector
14: end procedure
```

Speed and Accuracy Figure 3.7 shows the real-time camera capture with tracking overlay for multiple simultaneously tracked fabrics, both under normal circumstances and under various forms of occlusion.

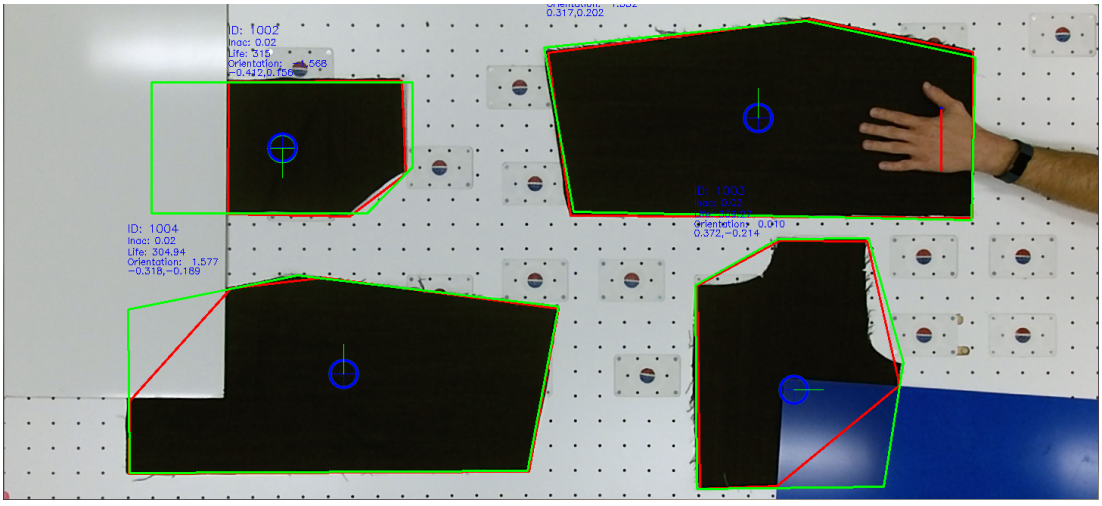
In practice on a reasonably powered computer (Intel Core i7-4790 @ 3.60GHz, 32.0 GB RAM) running Windows 10, the tracking algorithm operates on a full HD (1920x1080) frame at nearly the speed with which the frames are captured from the camera (15fps) and has been demonstrated simultaneously tracking 5 pieces of fabric, which is roughly the limit for number of fabric pieces that can physically fit within the camera frame at one time.

The reported position of the center of mass of the fabric is stable to less than 1mm, with the orientation (theta) stable to less than 1 degree.

Failure Cases The entire algorithm relies on the notion that the vertices of occluded polygon must lie either on a vertex or an edge of the target polygon. This is enforced in

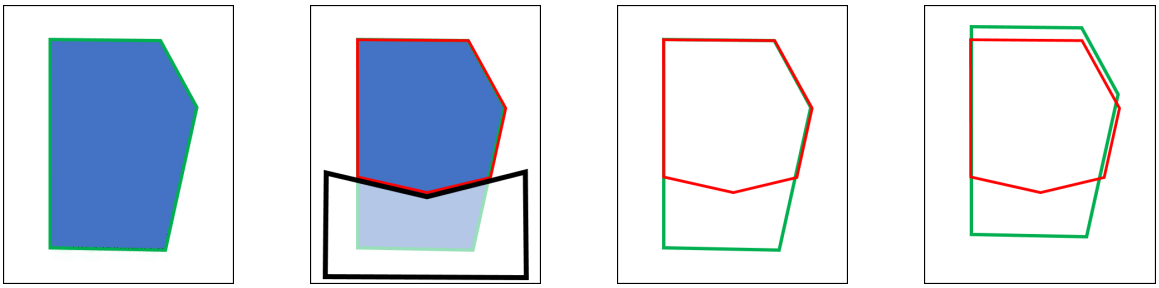


(a) Active system showing fabrics with their status info



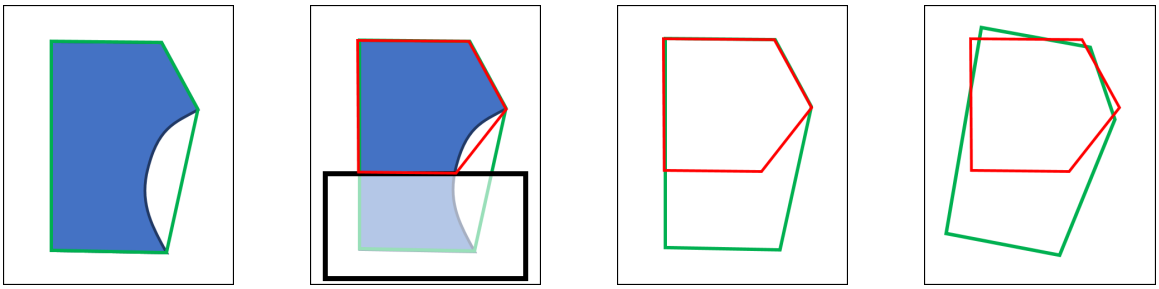
(b) Same fabric pieces under various states of occlusion

Figure 3.7: Simultaneous template tracking of 4 separate fabric pieces with their observed boundary (red) and predicted boundary (green)



(a) (b) (c) (d)

Figure 3.8: Tracking error case in which a fabric is occluded by a concave polygon. (a) Fabric (blue) with convex hull template (green), (b) Applied occlusion (black) with observed convex hull (red), (c) Observed vertices (red) with ideal template position (green), Actual position of template after running ICE2 tracking



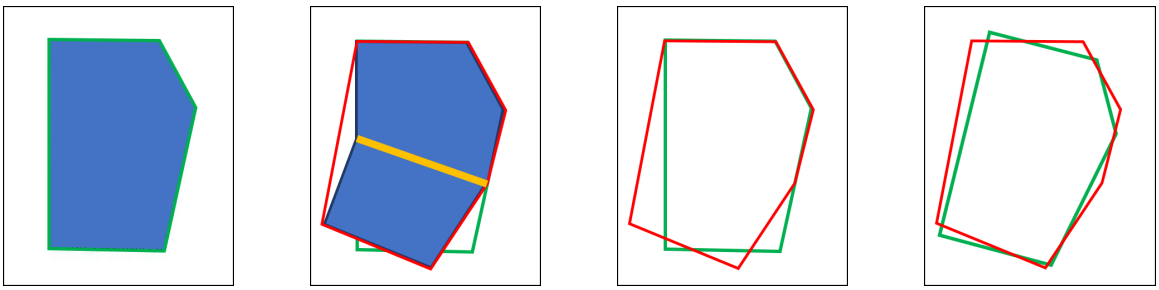
(a) (b) (c) (d)

Figure 3.9: Tracking error case in which a fabric with concave features is occluded. (a) Concave fabric (blue) with convex hull template (green), (b) Applied occlusion (black) with observed convex hull (red), (c) Observed vertices (red) with ideal template position (green), Actual position of template after running ICE2 tracking

practice by using the convex hull around any occlusion, however, in some cases in which the fabric is occluded by a concave shape, the convex hull may add an additional vertex on the observed shape which would lie completely internal to the target polygon. Depending on the placement, this additional vertex will generate a new error vector and act pull the observed polygon away from its ideal position (see figure 3.8).

A similar failure occurs for concave fabric pattern that's been occluded, as the new edge vertex is internal to the original convex hull (see figure 3.9).

Shape Deformations Part of the use of this algorithm is for identifying and correcting wrinkles internal to the fabric. However, with an excessively deformed fabric, the observed



(a) (b) (c) (d)

Figure 3.10: Tracking error case in which a fabric with concave features is occluded. (a) Concave fabric (blue) with convex hull template (green), (b) Applied occlusion (black) with observed convex hull (red), (c) Observed vertices (red) with ideal template position (green), Actual position of template after running ICE2 tracking

vertices are placed far from their expected position and result in erroneous matches, again because the base assumption of observed vertices lying on either the vertex or edge of the template polygon (see figure 3.10).

3.4 Wrinkle Detection

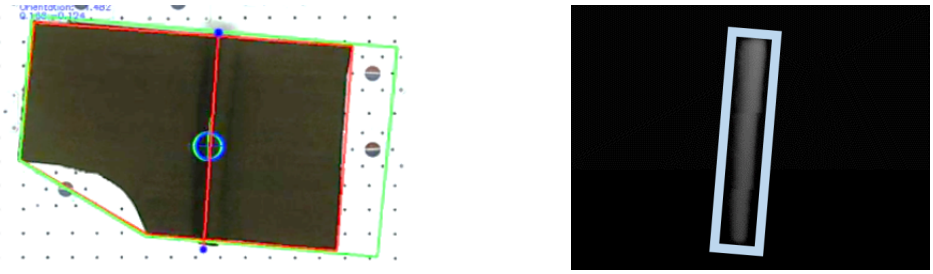
3.4.1 Algorithm

The wrinkle detection algorithm builds on the modular structure that is pervasive in the design of the distributed actuator system and builds on the template matching algorithm used for tracking fabric. The detection algorithm is designed to extract the minimal amount of information that describes the wrinkle sufficiently for the control system to take the appropriate mitigation action, not just for the improvement of processing speed, but also to minimize communication overhead between processing components.

Simplifications/Assumptions

While stated previously, it bears enumerating some of the assumptions that lead to the winkle detection and control algorithms provided here.

For hardware, the algorithm requires both a color and depth image that are correlated and position-matched for the proper extraction of wrinkle information. The resulting cor-



(a) Tracked fabric with the wrinkle and expected boundary identified

(b) Depth-map view of the same fabric shown with the minimum area rectangle enclosing the wrinkle

Figure 3.11: Wrinkle tracking uses a combination of the color vision template matching and correlation with the depth image

relation can be seen in Figure 3.11.

Under the assumption that this detection algorithm will be used with a distributed actuation system with some minimum feasible distance between actuators, there is a lower bound on the required fidelity of the wrinkle control. In practice, generalized information, or a major ridge line, describing the wrinkle is sufficient for control.

This algorithm relies on the existence of a pre-existing fabric tracking and template matching algorithm (described in this Chapter, Section 3.3) to provide localization information about the fabric and to minimize additional detection requirements, which was detailed in Section 3.3.

Detection

The wrinkle detection algorithm is designed from the ground up for maximum efficiency by examining and keeping a record of a minimal number of pixels and relies on the prior existence of the fabric identification algorithm.

To start, the depth sensor is calibrated to the table and a minimum tolerable wrinkle height set (5mm for initial testing). Upon detection of a fabric in the color camera scene (Figure 3.12a), retrieve the bounding box provided by the detection algorithm and crop the depth-map greyscale image to this bound area (shown in Figure 3.12b). This significantly decreases the pixel area within which to search for wrinkles and becomes the main input to

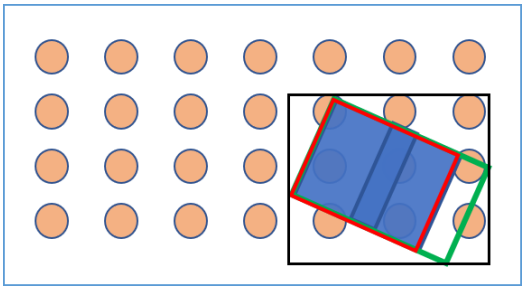
the wrinkle detection.

Figure 4 shows the overall flow-chart for wrinkle detection with the cropped depth image.

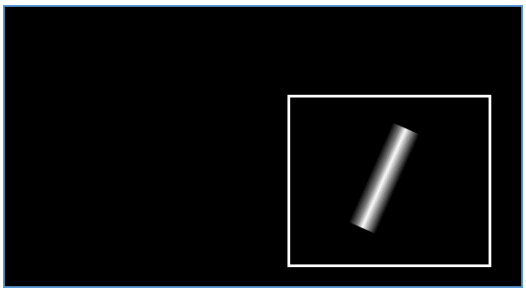
First the algorithm uses general blob detection on the greyscale image to find the largest non-black area, which corresponds to the observed wrinkle (if one is present). The blob detection algorithm can provide the minimum bounding rectangle (Figure 3.12c) from which the major axis direction and vertices can be extracted.

The major ridge line is defined by the effective height of the two endpoints of the observed wrinkle. Minimizing the number of pixels to traverse, the first and third vertex of the minimal rectangle can be used as origin points to step along the vectors perpendicular and parallel to the major axis (shown as green vectors in Figure 3.12e), only traveling 15% of the distance in the parallel direction and average the intensity of the visited pixels (shown as the orange box in Figure 4b). Note: Because pixel locations are in a grid, stepping along an arbitrary vector will result in some pixels being visited more than once or not at all. Rather than add complexity by tracking visited pixels, this error is accepted resulting in approximate height maps, which are sufficient for wrinkle control at the required resolution.

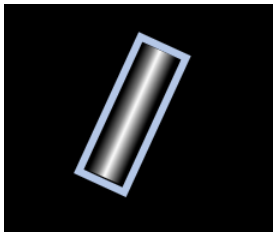
The averaged intensities in the outer 15% regions are then converted to an equivalent height using the calibrated intensity-to-depth conversion for the sensor and placed in space at the midpoints of the minor edge of the bounding rectangle (shown in Figure 3.12). This information now fully approximates the severity, general structure, position and orientation of the wrinkle, using two numbers to define a “major ridge line”.



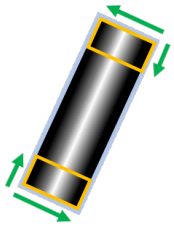
(a) Color camera view with observed fabric boundary in red, predicted fabric boundary in green and bounding box in black



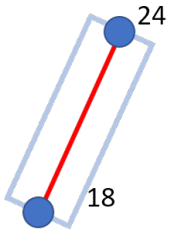
(b) Depth-map view with visible wrinkle and cropping region in white



(c) Determining minimum bounding rectangle around depth map



(d) Pixel traversal of ridge extrema with direction shown in green and examined regions in orange



(e) Major ridgeline defined by two approximate heights

Figure 3.12: Diagram of process for wrinkle detection algorithm.

CHAPTER 4
DISTRIBUTED CONTROL OF AN "UNACTUATED" ROBOT

4.1 Motivation

Referring back to Section 1.1.2, the goal of this system is to perform fabric handling and global transportation tasks within a realistic manufacturing environment. In a true manufacturing environment, high throughput and fault tolerance is desired. In the process of moving fabric, the flexibility of the material can result in deformations and wrinkles developing due to any kind of disturbance. The budgers provide a high degree of manipulation capability to compensate for these disturbances, however the higher number of actuators also increases the surface area for faults to occur. The flexibility and scalability needed to handle real world circumstances and still meet these goals is accomplished through the use of a decentralized hierarchical architecture to provide seamless, scalable, and continuous control described in this chapter.

4.2 Related Work

As this is an entirely new and novel actuation system there is very little work to draw on.

Referenced in 2.2, the work of Luntz [13][12][14] is the closest preceding analog to macroscale manipulation of a "virtual vehicle" using distributed actuators similar to the budgers presented here. However, all the control theory and practical designs used therein, rely on the computation of force balances developed through using the friction of slipping wheels against a rigid object and generating vector fields for desired motion. The budgers are designed to provide *no* slip and keep all motion in sync to avoid generating fabric deformations while also only using budgers as needed and on demand, so a completely different control system is needed.

Further, in the realm specific to fabric handling, there are no known investigations into distributed actuation. The vast majority research into the pick and place or gross free form manipulation needed for handling extremely flexible material that look for grasping points to unfold or properly drape a fabric [6][29]. In addition to being inapplicable, they also operate extremely slowly which would not fit into a typical manufacturing workflow. Schrimpf's work [10], on the other hand, does directly examine robotic operations for use in manufacturing, but is limited to local manipulation for tension and alignment of sewn fabric pieces. Practically speaking, these each dovetail into the budger research, which exists in the transportation niche between gross three dimensional handling and direct sewing.

4.3 System Architecture

4.3.1 Hierarchy of Control

The problem of control was examined here by flipping the perspective. Instead of building a machine that moves fabric, with the "machine" comprised of the actuation, sensing, and physical space, this paradigm examines an unactuated robot (the fabric) traversing an actuated environment (traditionally, the machine). This allows the system to be treated as two interconnected but distinct decentralized networks, one network being the environment and the other being the interacting fabrics. From this perspective, the robot communicates with and uses the resources of its environment to accomplish a task. The environment, here, makes up all the things that the fabric can interact with that don't belong to itself. This structure allows the environment to be broken into decentralized nodes, which can in turn communicate with each other, allowing the environment to be extremely flexible and scale arbitrarily without modifications to some centralized controlling hub. In practice, these interacting environmental nodes can take any number of forms. For the case of the budger, the budger table - consisting of the overhead camera (sensing), budgers (actuation), and physical table (space) - acts as the smallest decentralized environmental node.

Figure 4.1 shows the conceptual makeup of one of these nodes. Within the node, there

is a hierarchy of control using a the robot centric approach with the fabric as the primary entity. The node (table) is comprised of the camera and budger pool, encompassed as "shared resources" that belong to the environment. The fabric as a robot exists *within* this environment and communicates with the available resources in the environment to temporarily consume a subset of the resources. In this case, each fabric receives position information from the camera and uses a portion of the budgers to become a "robotic entity" that has all the necessary components for decision making and control. As the fabric moves, the resources available and in current use change, making the fabric a robot with a continuously changing configuration.

The control computations trickle down from the higher level as seen in Figure 4.2. The fabric, being the driving entity, computes its necessary control given its current state, which is then decomposed and provided to the budgers as an input where they each perform local control to provide the proper ensemble reaction. This is further discussed in Section 4.4. These controlled fabrics also each have their own goal states and may interact with each other to plan viable paths around each other and through the environment (continued in Section 4.6).

At a larger scale, each table node can be linked to neighboring tables to scale the machine to a full warehouse, making up the decentralized environment. This structure allows the computation to be broken down into sizeable chunks. The more complex multi-agent planning task only takes place within the confines of a single environment node, constraining the search space considerably and making solutions much more efficient. The crux of the full-scale system, then, is the ability for fabric to pass between environmental nodes to allow for arbitrary global locomotion.

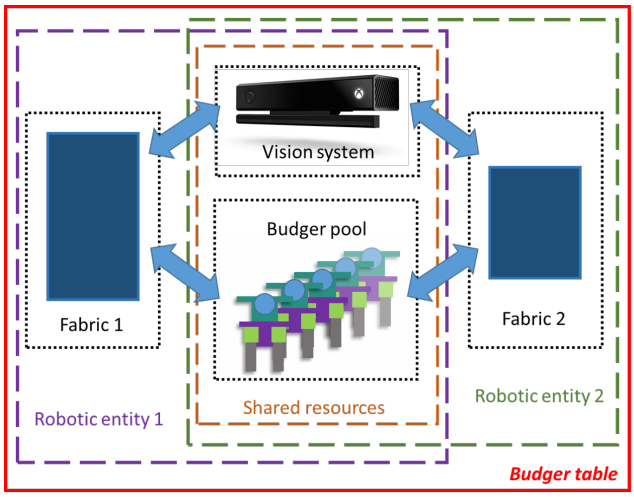


Figure 4.1: Control architecture with multiple fabrics sharing the same control and feedback resources.

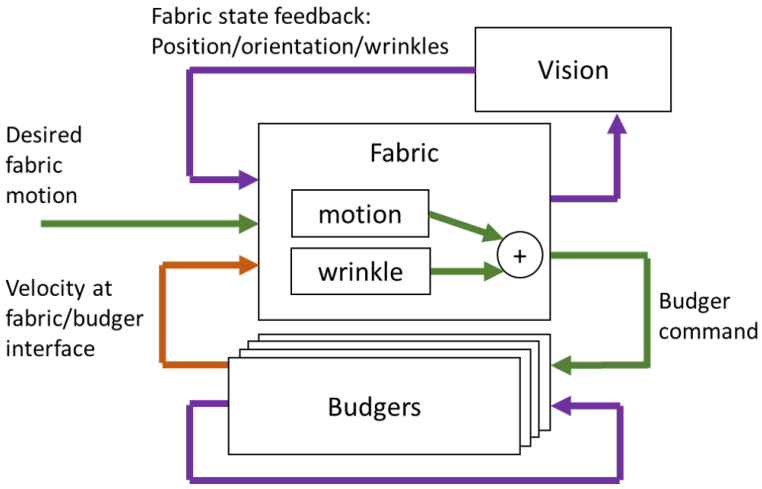


Figure 4.2: Overview of the trickle-down control and feedback loops present in the fabric locomotion.

4.3.2 Inter-table Handoff

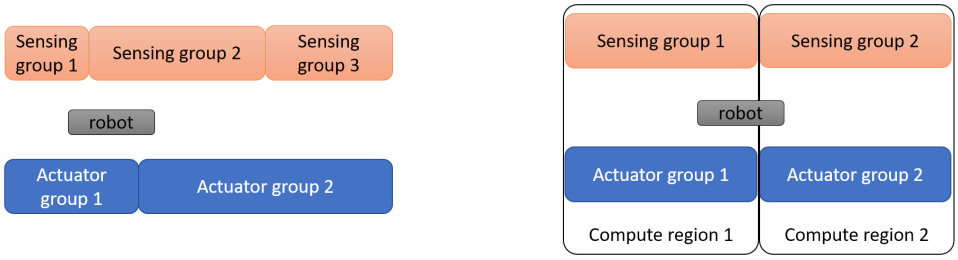
Motivation

As just described, the control of the "robot" takes place across an environment in the form of a network of nodes that make up the sensor and control capabilities provided to the robot. The environment itself, like the robots, is decentralized and comprised of individual non-overlapping sensor and control nodes. Each of these nodes have some level of computation and intelligence that make up the "smart" environment. When reaching a boundary of a node, there must be some form of coordination among the robot and the neighboring nodes to allow for a seamless transition as the robot physically spans the two regions and is using resources (sensing or control) from two nodes simultaneously. This seamless transition of material between decentralized nodes underpins the entire architecture that allows for the system to scale from a small local machine to the global routing infrastructure of essentially arbitrary size.

Specific to the budger system discussed in this thesis, the budger table is the environmental node and the fabric is the robot. In the discussed architecture, these tables are both physically and electronically linked to neighboring tables. Physical connections are those over which fabric can traverse (eg. a shared edge among two adjacent tables), while the communication connections can be more extensive but contain *at least* the same physical neighbors. The following section details the protocol that is introduced here to manage the physical handoff between nodes, first in the general case, and then as it applies to the budger system.

Protocol

Overview of the General Case Across all possible realizations of the unactuated robot (or underactuated, or sensorless, etc.) that relies on the local environment for some aspect of its sensing or control, the robot must coordinate three primary aspects:



(a) Most general case with each portion of the environment existing on its own with arbitrary boundaries.

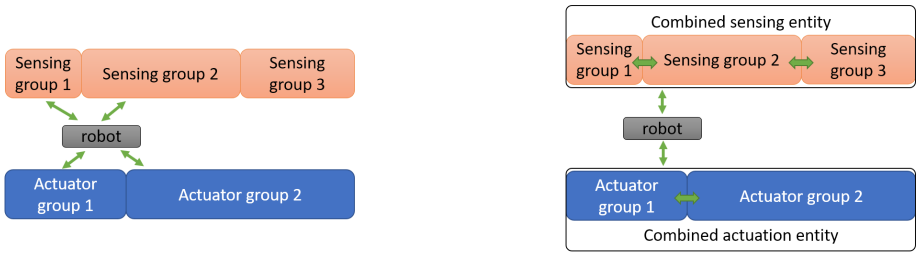
(b) Sensing and actuation portions of the environment are aligned and can share compute resources (as is the case with the budger table).

Figure 4.3: Comparison of possible environment states for robot handoff to handle.

- space
- sensing
- actuation

In the most general case, the different aspects may each operate within their own systems (or multiple systems), each with non-aligning boundaries. For example, there may be multiple sensor arrays, a set of cameras as well as lidar, whose regions of sensing overlap in an irregular pattern (see Figure 4.3a). In this most complex scenario, the robot would handoff between environmental nodes of different types at different times during traversal, which obviously requires a higher message throughput than the alternative in which each of the node types coincide spatially and computationally (shown in Figure 4.3b).

In either case, there are two primary ways in which to organize the protocol, either environment-centric or agent-centric. In the environment-centric approach, the nodes of each environment type manage the handoff amongst themselves independently and always provide pure information/control to the robot directly. In the agent-centric case, the robot manages all handoff information internally, and acts as a coordinator for maintaining up to date information by sending and receiving messages to the various neighbor nodes. These are each depicted in Figure 4.4. The former is more efficient with messaging and potentially allows for a faster refresh rate but requires the duplication of a lot of computational assets,



(a) Agent-centric: the agent has its own compute power and coordinates handoff internally by talking to all nearby environment resources.

(b) Environment-centric: environment resources coordinate handoff internally, so robot communicates with only one resource to receive full information.

Figure 4.4: Comparison of different handoff architecture types.

whereas the latter "centralizes" the handoff management at the cost of increased message passing. These are both discussed in more detail in the following section in direct relation to the budger system. Both approaches still use the same general protocol.

Returning to the three primary aspects, space, sensing, and actuation:

Space - It is assumed that the robot is a physical entity, and therefore takes up physical space inside the environment. Before entering into a new node region, the robot must check with the node to ensure that there is sufficient space to contain and maneuver the robot through the node so the first portion of the protocol relays information to the new node to check if it is able to accept more traffic. If no affirmative response is received, the robot either waits or tries to re-route. If the new node does accept the robot, the robot is free to cross the threshold between nodes, and the motion proceeds. During this period, the robot will be physically inside two (or more) nodes at the same time, which is where coordination is necessary among the remaining two aspects.

Sensing - Since it is assumed that sensing regions do not overlap, only a portion of the robot is visible within each node during the transition. The sensing data needs to be shared between the two nodes such that their combined information continues to provide an accurate representation of the robot. Ideally, this combination would be through efficient and minimal communication, in which they can cooperate on generating combined sensing without the need to share *all* their information. A specific instance of this in conjunction

with the template matching algorithm (Section 3.3) is discussed below.

Control - The physical robot can be shared across two separate physical control regions. The two regions must negotiate a desired instantaneous motion to impart to the robot and then generate the necessary control output. This may take the form of both (all) nodes receiving or sharing the current path or desired waypoint, the type of control computation along with specific gains, any limits due to saturation of the control output, and any other control modifiers and limitations such that together the two nodes agree on the same "action" for the robot at any given time.

The following section discusses this protocol in further depth using the budger table as the basis for a concrete example.

Budger Table The "fabric-as-a-robot" scheme presented for the budger table is a unique case in which the robot also does not possess any of its own computation or communication capability and therefore lives as a software representation of the physical object. Where this software representation lives depends on whether the system follows the agent or environment-centric approach. With the present architecture, the spatial, sensing, and control boundaries are all designed to align with the budger table as the decentralized environmental node (such that all three resources share the same boundary as in Figure 4.3b). For this reason, the agent centric approach was chosen, with the fabric representation living in the table control code. The robot existing as a software entity alters the protocol a bit in that instead of the robot straddling a boundary in the environment, it has to be duplicated across the nodes and the protocol takes on the additional task of keeping these two copies in sync.

Returning once more to the three resources, with the budger table these become:

- space: unoccupied physical region of the table at the interface with the delivering table
- sensing: position (x, y, θ) and wrinkle state information delivered by the vision sys-

tem discussed in Chapter 3

- control: individual budger actions that collectively move the fabric in the desired manner

As with the general case, the protocol begins with the spatial negotiation. As a fabric moves from Table A to Table B, A messages B with the size and shape information for the fabric and asks if there is room for B to accept this fabric at their shared physical interface. If B does accept, A then sends B the full representation of the fabric (including it's destination and control parameters) along with the known position and wrinkle state information. Each table then operates parallel and (nearly) identical copies of the fabric carrying out the same operations and maintain parity through communication. These copies are essentially identical, but because the two tables are fully independent, their refresh rate may be different. So while position information *is* continuously synchronized (every update cycle), at any given instant, there may be a small discrepancy between sensing and control state between the two tables. With the two copies in place, the sensing and control protocols proceed simultaneously.

Sensing: Initially, the sensing on B can't "see" the fabric, but table B continues to receive updates on position from A as A in turn receives it from its local vision system. B uses this information to "seed" its vision system as a starting point for the template matching algorithm. Since the algorithm stably locks on to the vertices and edges, by using a seed location, the vision system will know which shape to look for and be able to provide a stable location, even when only a small portion of the fabric becomes locally visible. (For further details, see Chapter 3, Section 3.3.) The protocol proceeds with A and B sharing their observed positions of the fabric. While B hasn't yet observed the fabric, it returns "null" and uses the position from A to continue seeding its vision system and A continues to use only its own position information. Once B sees a portion of the fabric, it registers this position and begins sharing it back with A. The two positions are each locally combined by any chosen technique (for example, a Kalman filter could be used with two inputs and

varying uncertainty). This brings up another benefit of the template matching algorithm in that it always returns the weighted error vectors of the resulting match which can be used as a consistent "score" of the system accuracy across tables in order to keep the combined representation weighted toward the most accurate observation. In the iteration presented in this thesis, the positions are combined via weighted average in which the weight is determined by the returned error vector, a ratio of the observed fabric area to the total expected area, and the time since the last message of the neighboring table. This weight is defined as $w = (1 - |err|) * (\frac{A_{obsvered}}{A_{expected}} * e^{-\frac{\delta t_{last}^2}{\tau}})$, where err is the average weighted error returned from the template matching (between 0 and 1), A is the fabric area (observed and expected), and δt_{last} is time elapsed since the last sensor response, with τ being a tunable scaling factor to account for expected sensor response rate. Such a weighting ensures a smooth and stable representation of the fabric during the handoff and seamlessly transitions from one table to the next. By the end of the handoff, the weights must be 100% weighted toward table B.

Control: The shared control is accomplished through minimal communication with the assumptions that both copies of the fabric representation use the same control scheme and feedback weights as well as the same feedback information. The latter is ensured by the sensing protocol just discussed. The control information can be weighted based on a global control mandate (in which all tables use a pre-agreed upon scheme and weighting) or, more flexibly, the information can be specific to each fabric and shared via the protocol during the initial negotiation and duplication. With essentially identical feedback and control calculations, the two fabric instances can generally run in parallel without communication and expect to output actuation for compatible fabric motion.

However, communication is still needed to handle any modifications to the expected control output. As mentioned further in Section 4.4.4, the active group of budgers has their speeds collectively scaled to avoid any budger exceeding their maximum speed. When a fabric spans multiple tables, the adjacent tables will perform this scaling independently, so

the expected scale factor for the table is added to the protocol and shared with the neighbor. This requires at least one message each way per control computation cycle. The scaling algorithm for each table is then modified to use the minimum scaling factor between its own and its neighbors.

Handoff completion: Once the originating table, table A, no longer sees any portion of the fabric, it assumes the handoff is complete. Likewise, table B does the same when it sees the full fabric. One last set of messages is exchanged to confirm the completion of the handoff in which table A alerts B that it no longer has the fabric and wishes to delete the instance. When B responds with an affirmative that it has full control, A can then remove the fabric from its set.

Implementation

The handoff protocol, as just discussed, is designed to operate between two budger tables with their own budger set, camera, and self contained computation. However, there is only one prototype table which adheres to the architecture described in Section 4.3.1. To implement and test the protocol, the table was virtually split into two along the y dimension at x = 0 (the middle of the table). Since the vision code and control code operate separately with one communication pipe between them, they both had to be separately split. This made the architecture more akin to an environment centric approach, so some modifications were made to the protocol. In this case, each virtual table is small enough that the initial negotiation is simplified to whether or not there is any fabric on the table, but in testing with a single fabric, can be ignored entirely. The vision program uses the same processing structure, but when an image is received, the image is split down the middle and each one is separately processed. Similarly the control program already uses a virtual "workspace" that acts as the table, so this was duplicated with half the budgers placed in each. The protocol operates internally to each program, so the vision program internally handles the vision protocol and then generates a single message output describing each half of the table. The

control program receives this combined message and internally splits it to send into each control workspace. Though the message-passing is slightly different and the tables are not entirely separated, the proof of concept performance is expected to be representative of a full scale system.

Performance

Using the method described above, the camera frame and table were split into two left and right half-tables and handoff simulated between them. Figure 4.5 shows the resulting qualitative tracking performance of a piece of fabric manually manipulated to pass between the two frames. The red outline shows what is visible to each table while the green outline shows the expected position for each table. What is most striking is that this appears nearly identical to the standard table vision system with the exception of the red line down the middle (the shared border between the tables) and some minor position error in 4.5e due to the curvature (which is a possible and expected error as seen in Figure 3.9). Typically these kinds of errors are offset by the sharing of information as they skew the results of the template matching algorithm in the opposite direction for either side. In this specific case, however, the right side is the sole driver of the algorithm. Generally speaking, these errors are minor and short lived. Figure 4.6 shows the time history of the measured and shared position information for the same handoff as Figure 4.5. Here it can be seen the regions on both the left and right where the fabric wasn't visible and the position was purely defined by the opposite side. Even when the measured position shows a moderate error, as is the case for the y and theta values when the left side loses and regains the fabric (around 36s and 40s mark), the shared position remains stable. While this is used as a proof of concept and further tweaking of the sharing algorithm is possible, it is helpful to note that these errors are small enough to not perturb the transition as generated by the budgers. In a test of this capability, the fabric was commanded to follow a series of waypoints that require the fabric to do a 180° turn while transitioning through the center (the hardest possible task

to handle). Errors were generated during that difficult transition, but qualitatively during this maneuver the fabric still managed to hit its target position and continue on to the next waypoint. Ultimately, this method shows that inter-table handoff is a viable solution to scaling the system to much larger sets of budger tables.

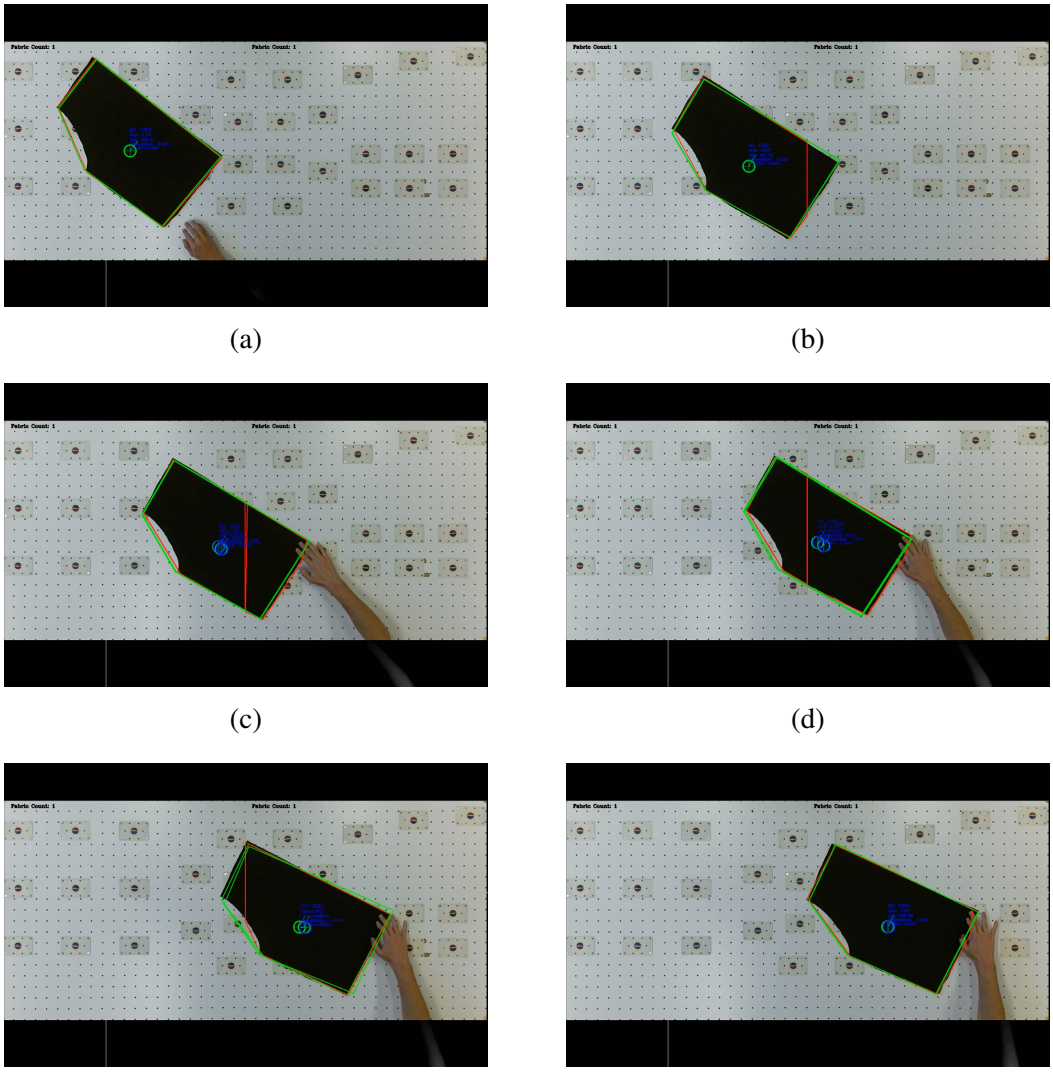
4.4 Control of Fabric via Budger Allocation

4.4.1 Resource Claiming for Locomotion

Described in 4.3.1, the budgers are considered an exclusive resource for use by the fabric. The budgers are the only source of actuation provided to the fabric and can only be used by one fabric at a time. For increased throughput, it is desirable to have more than one fabric active on any given table, therefore a system is needed to allow for these budger resources to be allocated between the active fabrics to ensure budgers are only in use by one fabric at a time *and* that they are only claimed and active when needed.

With the table the arbiter of local resources (vision and budgers) and the fabric acting as the controlled agent the resource allocation takes place as a "conversation" between the fabric and the table. In the case of vision resource, this is essentially an infinite resource as the vision system can view the entire table at all times and maintains position information for all fabrics simultaneously, so the fabric can simply request its position and receive that information. Likewise, any fabric can do the same at any given instant. Since the budgers are exclusively assigned to one piece of fabric, this conversation takes place as a request for access and subsequently granting (or not) exclusive access to those requested budgers.

Within software, the system consists of objects that mirror the physical environment. The table contains a list of all "live" and available budgers called the budger pool. On each update, the fabric entity requests and updates its position and then determines which budgers it needs. The table then checks the budger pool to see if these budgers are still available and releases those that are available to the fabric and removes these budgers from the availability pool. With the "ok" from the table, the fabric then directly adds the budger



(a) (b)

(c) (d)

(e) (f)

Figure 4.5: Snapshot images during manually transitioning fabric between sides of the table.

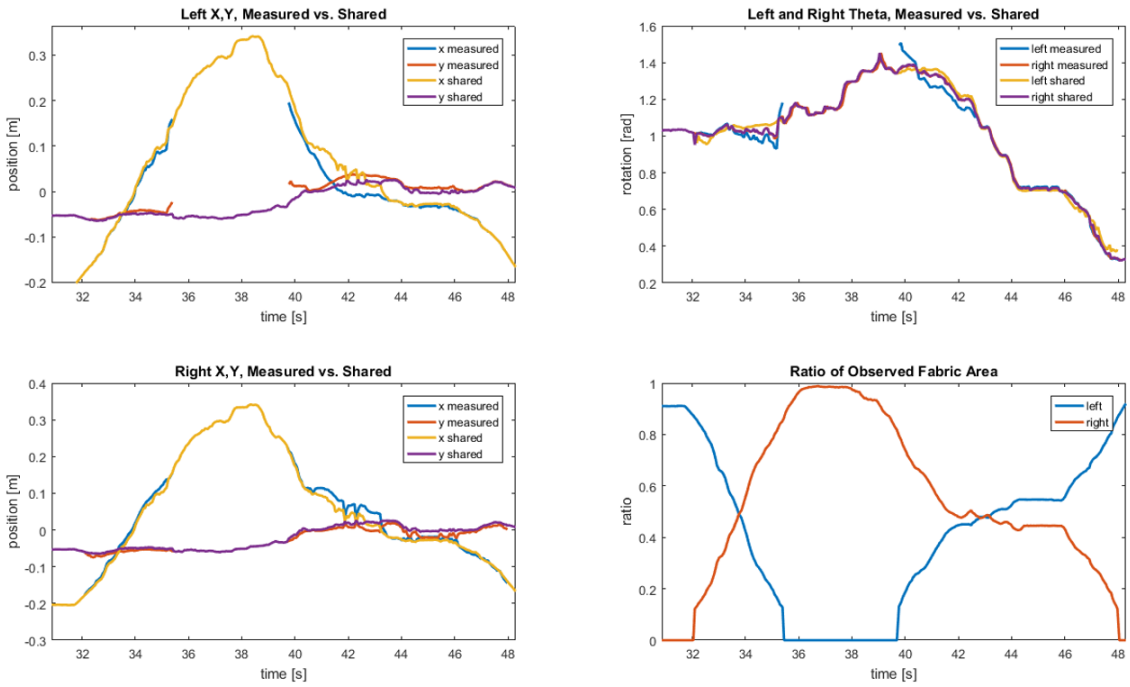


Figure 4.6: Recorded time history demonstrating the shared position stays stable through transitions between two sides of the table (split camera frame). This is the same transition as is shown in Figure 4.5

objects corresponding to the released budgers to itself. There are two possible scenarios at this point, either all the budgers requested were released to the fabric or some were unexpectedly unavailable. In the case that that some are unavailable and the the fabric claims an incomplete set of budgers, the fabric generates an error state and halts until the error is resolved. In the case that all budgers are returned, the fabric treats these claimed budgers as if they are its own omni-directional wheels and computes it's control according to it's desired motion and using the known locations of the budgers relative to it's current position. As the fabric moves throughout the table, it will then release budgers no longer in its sphere of influence back to the table and alert the table of the change. The table then returns these budgers to the available pool and puts them into an inactive state, ready for use by another fabric.

Now, to backtrack a bit, what was meant by which budgers a fabric "needs"? In the absence of planning (which is further discussed in Section 4.6), budgers are claimed on a first come, first serve basis. The budgers a fabric *needs* to control in any instant includes all the budgers underneath the fabric at that instant as well as any budgers expected to be contained in the fabric boundary between this instant and some forward prediction period. During this prediction period the fabric is projected forward both along the desired path, as well as the actual path based on its current translational and rotational velocity (to cover scenarios where the fabric is being moved by an external, unobserved/uncontrolled entity). Initially one might expect this prediction period to be the length of time until the next possible update. This is the minimum prediction period, however, because the budgers have a non-zero reaction time, the forward prediction time needs to include the amount of time it will take to move an inactive budger from any state to that required by the fabric by the time it arrives. In practice, the budger ready-time is roughly an order of magnitude larger than the feedback and control update rate for the fabric, and so it governs the prediction period. The longer this time is, the larger the area of budgers claimed by any one fabric, reducing throughput of the table, and increasing the uncertainty in the actual position of the

fabric, reducing performance. This, then, drives the requirements set forth in Section 2.3 for a fast point to point direction move for the budger. In testing, the new budger averages about 250ms, so accounting for margin of error, the system currently uses 0.5 seconds for this prediction window.

4.4.2 Fabric Control with Position Feedback

The fabric now has position feedback from the vision system (Section 3.3) and actuators in the form of claimed budgers (Section 4.4.1), which is everything it needs for closed loop control. To simplify the control calculations, the fabric is treated as a rigid planar massless object with direct velocity and rotation input in the form of the coordinated budgers. Although this is not a strictly accurate representation of the actual physical fabric (it's neither massless, nor rigid) or the input (budgers are non-holonomic), this model is based on the assumptions that:

- there exists a continuous deformation-corrective action that maintains the fabric in its undeformed state at all times, such that it behaves as a rigid body
- there is no slip between the budger and fabric, so output velocity of the budger is mirrored by local velocity of the fabric at the contact point
- the motions required of the budgers for differentiably continuous motion of the fabric is itself differentiably continuous
- any motion that does not meet the differentiably continuous requirement, or any other disturbances introduced by mechanical of software errors, can be compensated for via the wrinkle feedback system.

Under these assumptions the fabric control model is:

$$f = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \theta \end{bmatrix} = f_{xy} + f_{\theta} \tag{4.1}$$

$$\dot{f} = \dot{f_{xy}} + \dot{f_{\theta}} = u = v_{budger} \tag{4.2}$$

Where f is the fabric state, which can be described as a sum of translation, f_{xy}, and rotation, f_{θ}, components. The dynamics are described by the fabric velocities, which are directly controlled via budger velocities. For closed-loop control the control input can use a gain, β_f, on the fabric state error, with a feed-forward term for the desired fabric velocities, f_d.

$$u = \dot{f_d} - \beta_f(f - f_d) = \dot{f_d} - \beta_f(e_f) \tag{4.3}$$

Such that the fabric dynamics become:

$$\dot{e_f} + \beta_f(e_f) = 0 \tag{4.4}$$

With the total error between desired position and expected position driven to zero for lag-free path following.

4.4.3 Computing Budger Control

The state of a particular budger i, b_i, can be represented as either a combination of the speed and direction or the x and y components of the spin speed.

$$b_i = \begin{bmatrix} s \\ \theta \end{bmatrix}_i \Leftrightarrow \begin{bmatrix} x_{spin} \\ y_{spin} \end{bmatrix}_i \tag{4.5}$$

The latter is more convenient for vector calculations while the former better represents the physical reality in which two separate motors each control the spin and orientation separately (ignoring the slight mechanical cross-talk in the gear setup discussed in Section

2.3.1).

As Equation 4.3 shows, the output of the combined motion of the claimed budgers should directly be the rotation and translation velocity of the fabric. As such the desired rotation and translation of the fabric can be decomposed into a required state of each claimed budger to create this motion. With the velocity decomposed into components (translation, f_{xy}, and rotation, f_θ), the desired state of a budger can be computed as:

$$b_{d,i} = \dot{f_{xy}} + \dot{f}_\theta \times r_i \rightarrow \begin{bmatrix} x_{spin,d} \\ y_{spin,d} \end{bmatrix}_i \Rightarrow \begin{bmatrix} s_d \\ \theta_d \end{bmatrix}_i \tag{4.6}$$

The translation component is directly applied while the rotation component is decomposed via the cross product of the rotation and vector pointing from the desired center of rotation to the budger, r_i. The result of Equation 4.6 is a vector in $\mathbb{R}^3$. However, since rotation is orthogonal to the plane of motion, the third element *must* be zero, so the third element is dropped (operation indicated by the right arrow), leaving a vector in $\mathbb{R}^2$ corresponding to the x and y components of the budger spin. And finally the state is converted to the controllable speed and orientation (indicated by the double right arrow). This computation effectively "coordinates" the claimed budgers and these desired velocities are then communicated from the fabric to the individual budgers.

The spin of budger i, s_i, and the orientation, θ_i, operate their own local controllers at the budger level:

$$s_i = u_s = s_{i,d} \tag{4.7}$$

$$\dot{\theta}_i = u_\theta = \theta_{i,d} - \beta_{\theta,i}(\theta_i - \theta_{i,d}) \tag{4.8}$$

Here, the brushless motor controller operates as its own closed-loop speed control, so spin speed control is trivial. The orientation control uses the same feed-forward, feedback controller as the fabric with direct speed input (due to the closed-loop hardware brushless motor controller) and position feedback via optical encoder with a suitable control gain $\beta_{\theta,i}$

selected to provide fast response and prevent oscillations.

To use the feed-forward component of the controller, the next position of the fabric is used and the calculation repeated to generate the rate of change in orientation. In practice, the budger response rate is sufficiently quick relative to the movement speed of the fabric such that the lag between desired and actuation position is negligible. This extra computation was deemed unnecessary and left off to reduce complexity, leaving the dynamics as:

$$\dot{\theta}_i = \beta_{\theta,i}(\theta_i - \theta_{i,d}) \tag{4.9}$$

which is still stable around the desired theta setpoint.

4.4.4 Control Modifications

Velocity Capping

One issue in dealing with distributed actuation for locomotion is control saturation, in this case in the form of a speed limit of the budger. Because the budger is disconnected from the fabric and the placement of the budger relative to the fabric is infinitely variable, it is not sufficient to apply hard speed constraints to the fabric straight away. The system isn't limited in overall speed of the fabric or overall rotation rate of the fabric, but individual point-wise speed at the interface between fabric and budger. Allowing one of the budgers saturate without compensating for the others results in uncoordinated budger motion which can deform the fabric. While it would be possible to separately scale the fabric rotation and translation to maximize the average budger speed, this would alter the expected motion. Instead, this system implements an overall motion scaler. As the budger velocity values are computed, the fabric compares the spin speed of the budger to its control limit and calculates the speed to limit ratio. This ratio is tracked for each budger and after all budgers claimed budgers are completed, if the largest ratio exceeds 1, all the budger speeds are scaled by the inverse of this ratio. The result is each fabric taking full advantage of the available budger capabilities while maintaining the expected fabric path. Note also that

this method allows for differences in capabilities of each of the claimed budgers (eg. in the instance where high speed fabric rotations are necessary, the outer budgers may employ faster motors for the spin, or may be geared to sacrifice accuracy for speed).

Spin Reversal

The budger actuation is treated as a vector with spin speed and orientation. For simplicity, the spin speed is generally treated as ranging between 0 and the max speed, which results in only one possible orientation (plus or minus any multiple of 2π). In this scenario, changing between an orientation of 0 degrees to 180 degrees requires rotating the budger a full 180 degrees when running it in the opposite direction would perform the same task. Reversing direction is a faster operation (depending on the distance needed to travel) *and* more importantly reduces wandering of the fabric in the event of an overshoot on destination. In the original case, if the fabric overshoots by a small amount (but still outside a deadband tolerance), the budger might have a command from 0.2m/s at 2 degrees to a command of 0.2m/s at 170 degrees. In switching between the two, the budger will rotate through the 168 degrees needed, but the spin will stay active during that period, introducing more error into the position. Instead, this system allows the budger speed to exist between plus or minus the max speed, and the orientation is computed as angle that is closest between the current orientation and either the actual commanded orientation or the commanded orientation $\pm$180 degrees.

4.4.5 Fabric Position Control Performance

With the position feedback in place a feedback-feedforward controller was implemented that allowed for positional waypoint (x, y, θ) control (in which a static position is specified and maintained) that was robust to disturbances as well as timed path control for coordinating a sew path with the automated sewing machine.

Figure 4.7 shows a time lapse in which the fabric begins in an occluded state and is

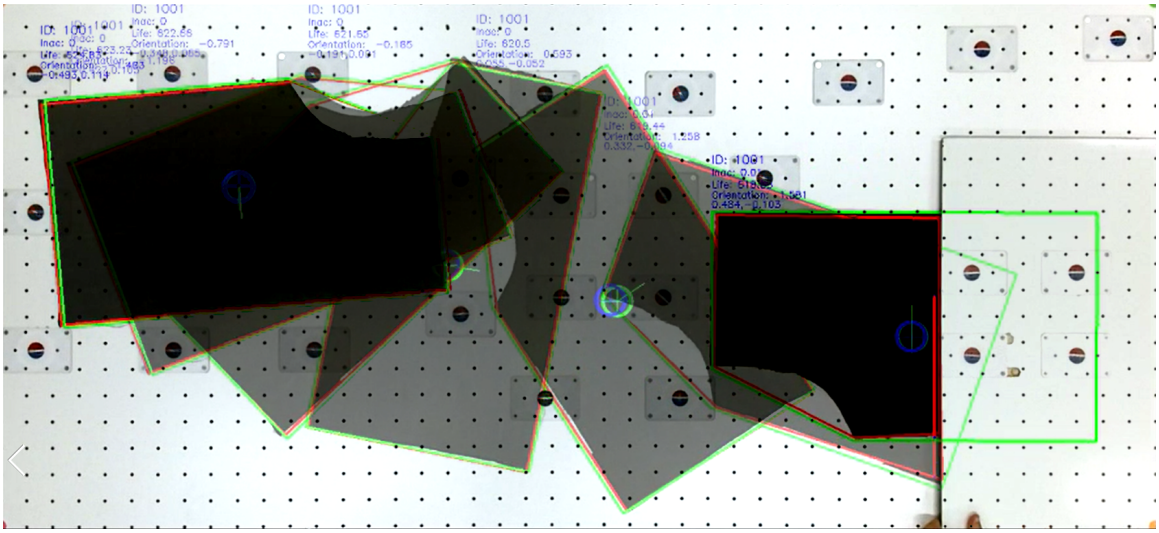


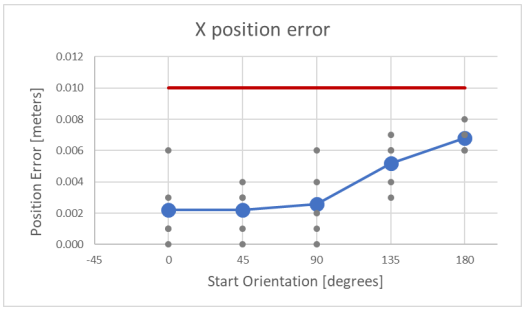
Figure 4.7: Demonstration of live tracking and control between two points on the table

controlled to rotate 180 degrees and translate across the table (each snapshot is roughly 1 second apart). Though difficult to show in pictures, the system qualitatively demonstrates smooth and continuous control and a strong robustness to occlusion.

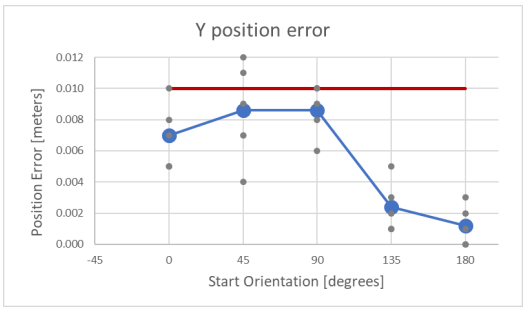
Although, the number of variable states the fabric can assume is inexhaustible, in order to test the control performance qualitatively a set of representative waypoint positioning cases were performed from various start conditions. The fabric was manually positioned in the center of the workspace at a specified orientation which increased from 0 to 180 degrees in increments of 45 degrees. Each starting orientation was then moved to a common final position on the left side of the table at the 0 degree orientation a total of 5 times. This was intended to provide a sufficient encompassing representation of combined and coordinated motion of the budgers under closed-loop control.

However, while it was stated in section 3.3.3 that the tracking algorithm was stable to 1 mm and 1 degree, the budger system is limited mechanically due to a number of factors, one of which is a limitation in the minimum speed at which the budgers can be driven. As a result there are hard-coded "success" tolerances of ± 0.01 meter and ± 3.6 degrees in which the system accepts that the fabric has reached a waypoint.

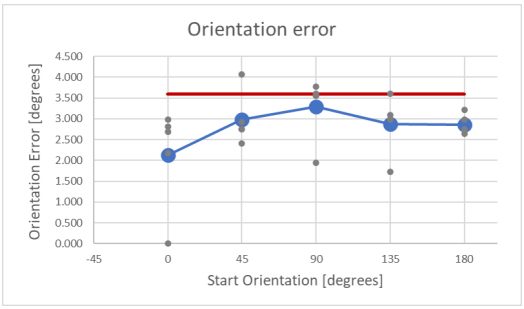
Figure 4.8 shows the results of the trails in with each result shown in grey and the



(a) error in x position



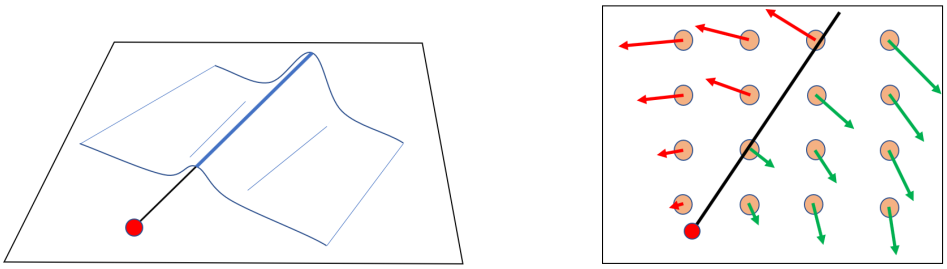
(b) error in y position



(c) error in orientation

Figure 4.8: Results of closed-loop fabric positioning test with varied starting positions

average error in blue with the bounding mechanical tolerance shown in red. To give a reasonable estimate of system performance, the trial was stopped and error recorded roughly 1 second after reaching the target location in order to avoid the case where the waypoint tolerance wasn't met and the system would be given an excessive amount of time to reposition the fabric. This is how some trial cases (shown in grey) wind up exceeding the tolerance error (above the red line). The results show that on average the feedback does allow for the overall system to meet or beat the specified tolerance. Further, it's possible for the system to meet a tolerance of nearly 2 mm on x-y position under certain cases. With proper improvements to the brushless motor controller to allow for accurate driving at slow speeds, it's likely the positioning tolerance can be driven to parity with the feedback tolerance.



(a) Exaggerated asymmetric wrinkle with ridge line projected to a center point (shown as red dot)

(b) Diagram of computer optimal budger velocity vectors

Figure 4.9: Budger control computation relies on computing tangent vectors about the wrinkle "center"

4.5 Elimination and Control of Fabric Wrinkle State

4.5.1 Wrinkle Elimination

The original intention for the wrinkle control was to maintain flatness of the fabric in transit, thus validating the fabric control assumption of pseudo-rigidity (discussed in Section 4.4). This section discusses the foundational algorithm used for real-time wrinkle elimination, which is further expanded upon in Sections 4.5.2.

Using the major ridge line extracted from the wrinkle detection process discussed in Section 3.4 it is possible to formulate a closed-loop control around the no-wrinkle condition. Rather than simply have the budgers pull the fabric apart, by applying velocities perpendicular to the ridge line, the proposed system implements a single control law that can operate on arbitrary non-uniform wrinkles (example shown in Figure 4.9a). The amount of material to be "unfolded" to smooth out the wrinkle is proportional to the distance from the point at which a line projected along the major ridge line intersects with the table. The exact amount of material is irrelevant to the calculation as any offset in scale acts as a gain on the subsequent controller, which has its own tunable gain. Using these assumptions, the control calculation goes as follows:

1. In 3D space, start at the higher of the two wrinkle heights defining the major ridge

line and project through the lower point to intersect with the zero-height location and record this as the center point (shown in Figure 4.9a).

2. Generate a budger velocity vector by taking the cross-product of the unit vector in the vertical z-axis (out of plane) with the vector pointing from the center point to a budger location. This will generate a set of properly scaled vectors indicating speed and direction for each budger rotating counter-clockwise around the center point.

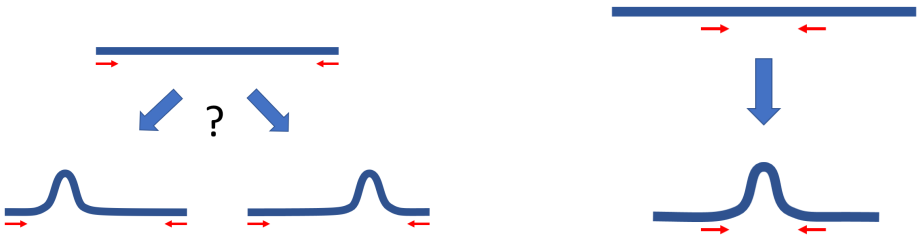
3. Test each budger's location relative to the ridge line. If it is in the left hemisphere, the velocity orientation is flipped to point away from the wrinkle ridge (these flipped vectors are shown as green in Figure 4.9b).

4. Finally, apply a suitable control gain proportional to the average height of the wrinkle (average of endpoint heights).

The result of this procedure is a closed-loop proportional controller for arbitrary wrinkles with a stable equilibrium at the zero-height wrinkle (or elimination of wrinkle). This controller, which amounts to a set of velocity vectors for the affected budgers, can be added to any existing fabric manipulation controller. Critically, the combination of control allows the wrinkle elimination control to take place while the fabric is in motion to maintain a planar pseudo-rigid-body, fulfilling the rigid-body assumption of the position controller. For instance, if moving in a straight line with a uniform wrinkle perpendicular to the direction of motion, the control action would result in faster budger speeds on the leading portion of the fabric and slower speeds on the trailing.

4.5.2 Wrinkle Creation

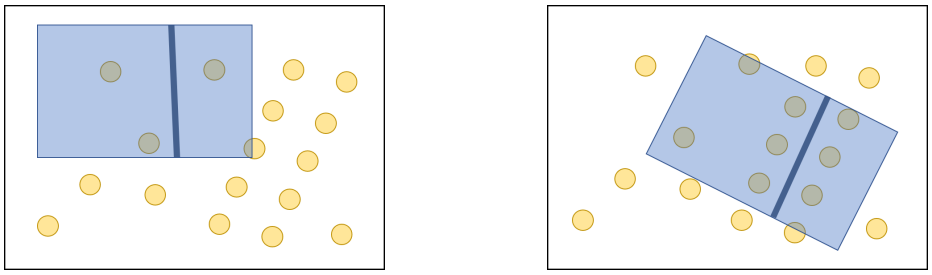
Although the primary and designed use case for wrinkle control is the elimination of wrinkles for maintenance of rigid-body motion, it is also possible to generate wrinkles, if desired. What follows is a discussion of the steps taken to enable this feature.



(a) With buckling forces applied far apart, there is greater uncertainty in the wrinkle creation location.

(b) By bringing the forces closer together, the wrinkle generation can be better controlled.

Figure 4.10: Creating a wrinkle in a fabric lying in a plane requires horizontally buckling the fabric, which has inherent uncertainty in the buckling location.



(a) Fabric poorly positioned for wrinkle generation with the given desired wrinkle.

(b) Ideal positioning of fabric.

Figure 4.11: For well controlled wrinkle generation, within a region there may be an optimal position to apply closely spaced forces along a line. Fabric and desired wrinkle (dark blue line) positioning relative to the available budgers matters.

Computing Feasible Wrinkle Generation Locations

Unlike wrinkle elimination, creation of a specific wrinkle can not occur at arbitrary locations on the table. Wrinkling in the case of the fabric on the table, where all motions and forces are planar, take the from of buckling. The exact position of buckling along a line of action between a force pair is unpredictable. Therefore, an optimal method of controlling buckling position is to bring the opposing forces as close to each other as possible, depicted in Figure 4.10. In the case of the planar table, where buckling will occur not just at a position but along a line, the ideal location of a desired wrinkle would be between and aligned with two rows of parallel budgers that are close together (Figure 4.11).

Of course on a real budger table, one could design the table to have a "wrinkle creation" zone to move the fabric to in order to create a wrinkle. However, this might entail traveling a large distance or placing an excess density of budgers throughout the workspace, which are both undesirable and in many cases infeasible. Take for instance the bunching of material while the fabric is passing through a sewing machine. In this case, it would be impossible to move the fabric to another location. Instead, the wrinkle generation should happen at the closest useable location, given the local budger layout and constraints on how tightly the wrinkle placement is relative to the fabric and how much movement of the fabric is acceptable.

These constraints take the form of a desired wrinkle line relative to the unwrinkled fabric template and tolerances on allowable rotation and perpendicular translation of that wrinkle within the fabric and rotation and translation of the fabric itself (with the fabric constraints superseding the wrinkle constraints).

The algorithm takes the form of two parts: the generation of ideal wrinkle locations and the application of constraints within those locations.

Finding Locations Similar to the template matching algorithm for vision (Section 3.3), the wrinkle locations are found using an iterative translational algorithm operating on weighted error vectors. In this case, the error vectors are between the budger locations and the current desired wrinkle placement. The weighted vectors apply a force balance to the wrinkle, generating a net "moment" and "force" to translate and rotate the desired wrinkle in space. The result is a wrinkle location oriented in such a way to divide the active budgers into balanced groups with a minimal distance between the groups.

What does a "balanced group" mean? Figure 4.12a shows an evenly split section of budgers in which there are an equal number of budgers on either side of the dividing wrinkle location, but the division is "unbalanced" in the sense that the budgers closest to the wrinkle would seemingly impart a local moment and, because fabric is deformable, pos-

(a) Unbalanced budger placement resulting in a moment and possible twisting of the fabric.

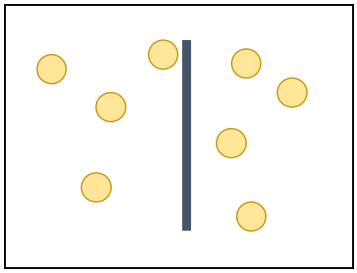
(b) Same budger layout with balanced wrinkle placement to reduce moments on the fabric.

Figure 4.12: Comparison of optimal wrinkle creation locations depending on budger positioning with budger motion shown pointing toward desired wrinkle state.

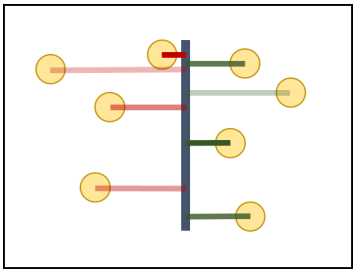
sibly generate a local twist. While this can't be entirely eliminated on arbitrary budger patterns, it can be reduced. Figure 4.12b shows the same pattern with a more balanced wrinkle location that reduces net moments. Using an iterative force-balance approach to generate positions for wrinkle creation inherently minimizes these moments and balances the division.

Figure 4.13 shows an overview of the algorithm in which, given a dividing wrinkle line, that line is translated and rotated into the optimal position keeping the budgers divided into those same groups. For any start location, the algorithm finds the local optimum. The space is then searched with random restarts to generate a set of possible wrinkle locations which can be scored via a suitable cost function.

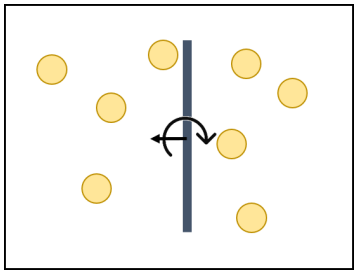
Applying Constraints To recap, constraints exist as limits on how far the fabric can move and how far the wrinkle can form from the specified position on the fabric. Both sets of constraints, on the wrinkle location and fabric location, exist in the same degrees of freedom (x, y, theta). Because they share this same space, the constraints can be stacked additively to generate the total allowable deviation from nominal for the wrinkle location. This takes the form where the full allowable wrinkle error region (err_{allow}) is defined as:



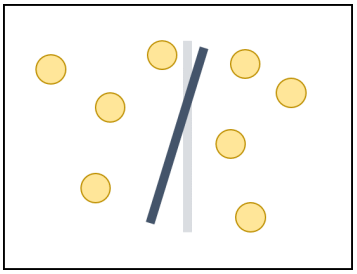
(a) Given a random assortment of budgers, the preferred wrinkle location is defined.



(b) Vectors are computed from the budgers perpendicular to the wrinkle and weighted based on distance (shown as darker for heavier weight).



(c) The weighted vectors are treated as individual forces and applied to create a moment and net force.



(d) The moment and force generate a small motion (original wrinkle location shown faded for reference), and algorithm repeats

Figure 4.13: Overview of the iterative algorithm for optimal placement for wrinkle creation.

Algorithm 5 Locally Optimize Wrinkle Location

procedure OPTIMALWRINKLE($v_{wrinkle}$)
  initialize R, T, err ▷ rotation matrix, translation vector, error value
  while $R, T \geq tolerance$ **do**
    for each budger **do**
      compute $p_{\perp,i}$ ▷ perpendicular vector from budger to wrinkle
      $err_i \leftarrow w\dot{p}_{\perp,i}$ ▷ apply weight to the vector
    end for
    extract T and R from set of err_i
    $T \leftarrow \delta_{translation}T$, $R \leftarrow \delta_{rotation}R$ ▷ apply δ scaling
    $v_{wrinkle} \leftarrow Rv_{wrinkle})$ ▷ apply small rotation
    $v_{wrinkle} \leftarrow T + v_{wrinkle}$ ▷ apply small translation
  end while
  return $v_{wrinkle}$ ▷ return modified wrinkle location
end procedure

$$err_{allow} = \begin{bmatrix} err_{x,allow} \\ err_{y,allow} \\ err_{\theta,allow} \end{bmatrix} = \begin{bmatrix} f_{x,allow} + w_{x,allow} \\ f_{y,allow} + w_{y,allow} \\ f_{\theta,allow} + w_{\theta,allow} \end{bmatrix}$$

with each component of the error the sum of the allowable error in fabric (f) and wrinkle (w).

This combined allowable deviation space determines the area for the random restarts on the optimization algorithm (Algorithm 5). In addition to bounding the start location, this space can be applied as hard constraints to bound the end locations through early termination of the algorithm. The constraint is enforced by ignoring the "force imbalances" that continue to push the wrinkle location beyond the constraint boundary. This means that it is possible for the algorithm to terminate with a remaining force imbalance, which can subsequently be used for the weighted scoring to chose between the generated wrinkles. In any case, the result of the modified algorithm is a set of possible wrinkle locations defined by x, y, theta that fall within the combined constraint area.

Because both of these constraints have overlapping degrees of freedom, as long as the selected wrinkle location does not fall exactly on the edge of this constraint region, the deviation of the wrinkle location from the original placement has to be divided between the wrinkle and fabric allowances. This can be achieved in any number of ways. In this case, it is chosen to be via a selectable weighting function in which the total deviation is a combination of wrinkle x, y, and theta and fabric x, y, and theta weights, such that the respective weights (eg. x wrinkle, x fabric) sum to one. So if

$$err_{actual} = \begin{bmatrix} err_{x,actual} \\ err_{y,actual} \\ err_{\theta,actual} \end{bmatrix},$$

then $\begin{bmatrix} f_x \\ f_y \\ f_\theta \end{bmatrix} = \alpha * err_{actual}$ and $\begin{bmatrix} w_x \\ w_y \\ w_\theta \end{bmatrix} = \beta * err_{actual}$ with $\alpha + \beta = 1$.

The exact weights are tunable to the needs of the application, but are left as 0.5 for each

in this instance, such that the relative allotments for each constraint is equally weighted.

Algorithm 6 Locally Optimize Wrinkle Location with Constraints

procedure CONSTRAINEDWRINKLE($v_{wrinkle}, err_{allow}$)
  initialize R, T, err ▷ rotation matrix, translation vector, error value
  for $j = 1$ to $n_{restarts}$ **do**
    $v_{wrinkle,j} \leftarrow R_{rand}\dot{v}_{wrinkle} + T_{rand}$ ▷ apply random rotation and ▷ translation within err_{allow}
    while $R, T \geq tolerance$ AND $v_{wrinkle,j}$ inside err_{allow} **do**
      for each budger **do**
        compute $p_{\perp,i}$ ▷ perpendicular vector from budger to wrinkle
        $err_i \leftarrow w\dot{p}_{\perp,i}$ ▷ apply weight to the vector
      end for
      extract T and R from set of err_i
      $T \leftarrow \delta_{translation}T$, $R \leftarrow \delta_{rotation}R$ ▷ apply δ scaling
      $v_{wrinkle,j} \leftarrow Rv_{wrinkle,j}$) ▷ apply small rotation
      $v_{wrinkle,j} \leftarrow T + v_{wrinkle,j}$ ▷ apply small translation
    end while
    record stats ($T, R, p, v_{wrinkle,j}$)
  end for
  $cost_{wrinkle} =$ score and weight stats
  $v_{wrinkle} \leftarrow v_{wrinkle,i}$ with $min(cost_{wrinkle})$ ▷ select minimum cost wrinkle
  return $v_{wrinkle}$ ▷ return modified wrinkle location
end procedure

Selecting Optimal Location The random restarts with the constraint-modified version of the wrinkle generation algorithm (Algorithm 6) result in a set of possible wrinkle locations from which to choose. As the context varies for what the wrinkle is intended for and what the constraints are, the selection of the most optimal location may be highly subjective. Some desirable criteria available for a cost function are:

- proximity - perpendicular distance of closest budgers to the wrinkle position. Lower is generally better, but too close would result in the budgers losing contact as the wrinkle is generated - which means there's a "sweet spot" for wrinkle generation dependent on the flexibility of the material in terms of ability to accurately generate a wrinkle exactly as desired.

- count - how well the budgers are allocated under the fabric. It is desirable to have a minimum number on each side of the fabric, which can be easily scored as the difference between the number of budgers on each side, with lower being preferred (lowest being 0, corresponding to each side having an equal number of budgers).

- deviation - how far the found wrinkle location is from the starting location. In general lower is better, but this can change depending on the needs of the operation. This is taken in concert with the constraints. It may be *allowable* for a large deviation, but still highly preferable for a low deviation, therefore there may be very relaxed constraints, but a high weight on this trait.

- balance - remainder of the force balance operation. While normally negligible, this could be a large factor in the constrained optimization. (Lower is better.)

In the existing system, this function is treated as a proof of concept, so the exact cost function combining these traits is somewhat arbitrary, and could (and *should*) change per the individual application. However, this particular implementation uses an equal weighted scoring system in which each attribute is calculated as follows:

- proximity - $|min(d_\perp - d_{\perp,ideal})|$, in which $d_{\perp,ideal}$ is the "ideal" distance (60mm in this case)

- count - $|n_1 - n_2|$ where n is the number of budgers on each side of the wrinkle

- deviation - angle: $1 - cos(\theta - \theta_{desired})$ and distance: $\left\| \begin{matrix} x - x_{desired} \\ y - y_{desired} \end{matrix} \right\|$

- balance - $F_{rem} + M_{rem}$, where F and M are the forces and moments calculated during the algorithm. (Note, although units don't match, the general magnitudes at the end of the algorithm are comparable, and can be combined easily.)

The scores for each location are normalized by the max and min of their respective attribute and mapped between 0 and 1. The normalized scores are then combined by weighted

average across the attributes for each wrinkle location (although in this nominal case all weights are equal) and the location with the lowest average is selected. This method gives qualitatively desirable results.

Generating and Maintaining Wrinkle Status

With the wrinkle generation location selected, the next step is to actually generate the wrinkle. This control piggybacks on the control scheme for wrinkle elimination. In the wrinkle elimination control, as restorative force is always added to move the budgers in a way that pulls the fabric apart from the wrinkle location. This is essentially equivalent to having a zero setpoint for the the desired wrinkle in a closed-loop control. Generating a wrinkle, then, involves inputting the desired wrinkle state, which generates a negative wrinkle depth (wrinkle observed - wrinkle desired). One tricky aspect, as mentioned previously, is that the wrinkle can not be created precisely since it's "buckling" the fabric and depends on the particular arrangement of the budgers and other properties of the fabric. The wrinkle location is chosen to optimize the likelihood that the wrinkle will generate in the appropriate spot, but it is not guaranteed. As such, the wrinkle generation has to allow for this uncertainty and tolerate some deviation from the desired location. This results in a two-state switching controller that switches between "make wrinkle" and "maintain wrinkle".

Create Wrinkle The wrinkle creation state is active when no wrinkle is detected by the wrinkle detection system (Section 3.4). In this mode, the desired wrinkle state (x,y, height for each endpoint) is set relative to the fabric and the optimal wrinkle location is continually computed as the fabric is moved into place. With the fabric in place (within a set tolerance) the wrinkle creation control is activated which will begin to drive the fabric together toward the chosen optimal wrinkle location (essentially opposite figure 4.9b). Barring a failure or extreme slip, this *will* generate some wrinkle within the region between counter-rotating budgers. Once a wrinkle of any kind is detected, the control is switched to maintenance of

the wrinkle.

Maintain Wrinkle When a wrinkle exists in the fabric, the only action the budgers can take is increase or decrease its size. The budgers have no actuation authority to translate the wrinkle within or relative to the fabric boundary. Consequently, in maintenance mode, the controller now ignores the desired position of the wrinkle and instead maps the desired heights to the nearest wrinkle endpoints. These desired heights are subtracted from the observed heights which creates the "actual - desired" pair that allows the closed loop control to proceed.

In the event that the wrinkle was generated outside of a specified tolerance, the desired wrinkle heights will not be mapped onto the existing wrinkle and standard wrinkle-elimination will proceed to remove the generated wrinkle. Once no wrinkle is visible, the mode switches back to wrinkle creation and the process starts over.

4.5.3 Limitations

This method of wrinkle control was initially designed exclusively for the purpose of maintaining flatness of the fabric, i.e. strictly elimination of wrinkles. While adding the creation option allowed wrinkle control to be possible, it comes with a few limitations. The largest issues are a result of the means of detection. Because the wrinkle detection (Section 3.4) uses the depth data from the camera, the "amount of wrinkle" is determined by the height. Beyond a certain height, which depends on the flexibility of the material, the fabric will fold over and only report an unchanging height even as the fabric continues to be compressed. Additionally, the detection method focuses on just one (the largest) wrinkle, which means that in generating wrinkles, there can only be one prescribed wrinkle location, and since that wrinkle is also limited in height, as just mentioned, the real-world utility of the system as it exists is limited. Another challenge is the strict placement of the wrinkle. The nature of the uncertainty of local fabric buckling (previously shown in Figure 4.10), makes very

tight placement impossible. Lastly, as will be discussed in the performance section, the motions needed to maintain the wrinkle are relatively small in relation to those needed to move the fabric along the table, which causes wrinkle maintenance while in motion difficult and may rely on implementing speed limits on the fabric motion, which is undesirable.

4.5.4 Performance

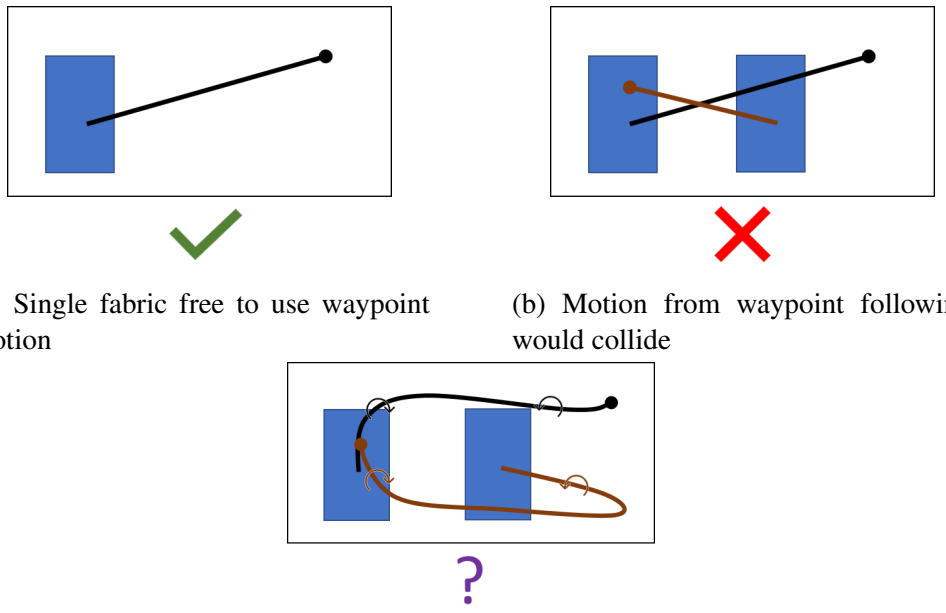
In practice, wrinkle detection is able to perform at real-time speeds, with nearly no additional delay to the already established tracking algorithm [30][31] (Section 3.3). On a reasonably powered computer (Intel Core i7-4790 @ 3.60GHz, 32.0 GB RAM running Windows 10), the combined template tracking and wrinkle detection operates at about 15 frames per second (very nearly the same rate as frames are gathered).

This system has been tested to provide extremely quick wrinkle/bunching rejection capability, and allow for continuous rigid-body control of various fabric sizes/shapes under large-scale positional transformations (rotations and translations).

In line with the limitations in the previous section, stable wrinkle creation was unreliable in execution. Although the fundamentals are present, and the generation of wrinkle location is successful, more work is necessary to provide stable wrinkling that can maintain the wrinkle during position changes.

4.6 Path Planning through an Actuated Environment

Under simple conditions in which only one medium to large size fabric is on a budger table, no planner is really necessary, with control governed by direct transition to the goal point or waypoints. However, as the table becomes more crowded or the fabric is small relative to the spacing of the budgers, the paths may need to be pre-planned (i.e. prescribed in advance) to ensure collision free motion without loss of control (Figure 4.14).



(a) Single fabric free to use waypoint motion

(b) Motion from waypoint following would collide

(c) Complex timed paths that include rotations could allow the fabrics to pass.

Figure 4.14: As table becomes more crowded, waypoints can generate collisions and path planning may be necessary.

4.6.1 Centralized Multi-agent Planner

Because the fabric "agents" don't have their own computation and are instead simulated on the local computer it was possible to use a locally centralized planner for the fabrics on the table, such that all existing fabrics on the table are simultaneously planned and then provided the resulting paths.

For the path planning algorithm, a rapidly expanding random tree (RRT) [32] was chosen for its ability to perform non-discretized, free-space exploration, efficient search of larger dimensional spaces, and a "built-in" ability to check variable constraints (eg. whether a fabric state would have enough budgers for actuation). RRT is a sampling based search in which a semi-random (possibly weighted) point in the search space is sampled and driven towards from the nearest expanded state by some finite amount. To simplify the computation, the robot dynamics were ignored based on the assumption of direct velocity control

via the budgers and a very low inertia of the fabric "vehicle". While the 2D (x,y) space is simple to search, to include rotations (theta) the search space was converted to a "time distance" space in which the dimension was divided by the desired speed of the fabric along that dimension in $\mathrm{m/s}$ or $\mathrm{rad/s}$. This allowed the search to use unit length steps in every direction. Because RRT does not guarantee an optimal solution and often results in "jagged" paths, the smoothness and optimality was improved with the implementation of RRT* [33]. RRT* adds computation overhead by re-computing the shortest path by re-connecting all the search branches local to a newly added branch, but this was deemed acceptable to provide smoother trajectories for the fabrics with less rapid switching required of the budgers. Lastly, RRT* is asymptotically optimal in that with infinite samples, the path will become optimal. To reach a "good enough" solution in faster time, the search was augmented with informed-RRT* [34]. Once an initial solution is found, informed-RRT* constricts the search space to an ellipsoid defined by the length of the minimal solution so far, which allows the solution to improve more rapidly and continuously reduce the search space. This implementation was further reasoning behind the conversion to a uniform time distance search space, to allow the uniform construction and sampling of ellipsoids within the space and still account for some constraints on the system.

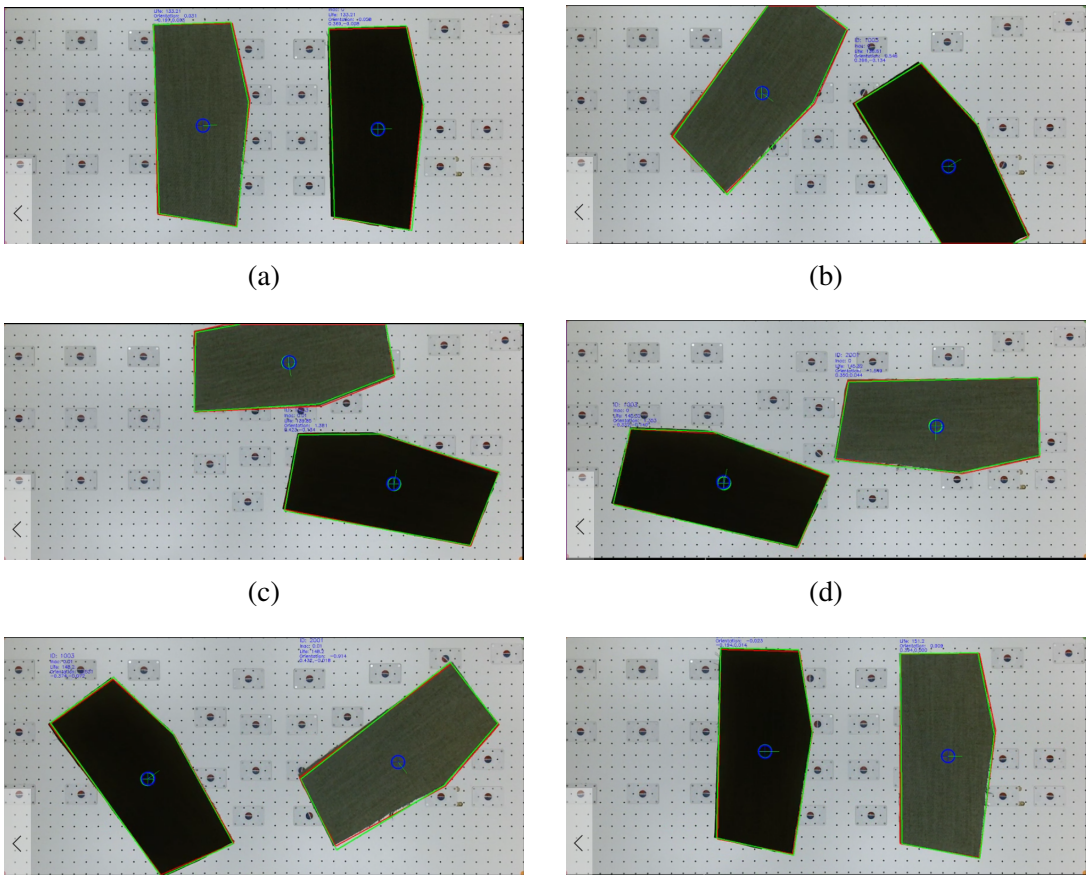
Combining states The normal search space for the RRT algorithm is three dimensional for each of the fabric state variables (x, y, θ). When doing a multi-agent search, the state for each agent are stacked into a $3n$ dimensional state, where n is the number of fabrics on the table. Once this state is converted to a time distance space, search proceeds in the same manner as with just a single fabric.

Including constraints One advantage of RRT is the ability to check constraints as new nodes are added to the search. In addition to standard collision checks, for the unactuated robot, this involves checking whether in its new position the configuration will still be controllable and visible by the environment sensing. In the present case, anywhere reachable

in space is considered visible. The controllability check is simplified to a check if the fabric has 3 or more budgers in contact with it, which is done by performing a point-in-polygon check of the local budger positions and the polygon positioned by the fabric's 3D state. Additionally, for the multi-agent case, interference between agents is also checked. For inter-agent collisions, the current $3n$ dimensional state is split into each of the agents' own states setting the polygon position of each fabric and are pairwise checked such that no vertex of any fabric lands inside any other fabric. If any of the checks fail, the searched node is discarded and the RRT algorithm continues.

4.6.2 Performance

Initially it was intended for the path planning to take place at a high enough speed that the planner could adjust the path during fabric motion. However, in practice and due to performance limitations of the developed code, it took on the order of 10s to generate a viable path - too slow for in-motion updates. Qualitatively, the generated paths were indeed viable and successful when used as pre-planners as a demonstration of the system's capabilities. Figure 4.15 shows a successful path planning task. In the depicted case, the two fabrics were positioned side by side and told to switch places. This demonstration shows the full possibilities of the system as the exchange involved tracking both fabrics simultaneously as they were occluded along the path, sharing of budger resources as budgers were only activated locally as necessary, and a fairly complex path with a large search space and narrow set of possibilities.



(a) (b)

(c) (d)

(e) (f)

Figure 4.15: Snapshot images of a position exchange of large fabric pieces using the central planner

CHAPTER 5
CONCLUSION

In starting with a pre-existing actuation paradigm for fabric control, this thesis extended that work through the introduction of the unactuated robot in an actuated environment to provide a viable and scalable fabric transport and manipulation system. This research fills a niche market for fabric manipulation that fits between existing, slow-moving fabric perception and manipulation research (eg. multiple serial arms with grippers to fold or unfold a garment) and high speed, high throughput mass production lines with regions of hard automation (eg. pocket sewer requiring fabric to be manually locked in place). The budger system targets flexible automation tasks in the newly emerging push toward low volume production of custom goods which results in high variability of single-layer fabric that needs to be grouped and routed to different sewing operations within a manufacturing facility. This was made possible by the combination of the individual contributions of the research into the actuation, sensing, and control of the unactuated fabric "robot":

- The budger mechanics were improved to provide significant performance improvements while also increasing serviceability and feasibility for real-world manufacturing applications.

- The capabilities of the budgers system were quantified in terms of frictional strength available to locomotive force to fabric under various conditions both with and without fabric floatation and budger suction.

- An algorithm was developed for a means of robust visual tracking for flexible materials in primarily 2D space that provides stable positioning under wrinkling and occlusion through and extension of the iterative closest point algorithm that uses a low number of points and the edges between them.

- Wrinkle detection and control was made possible by introducing a combination of fast wrinkle detection from the depth image with a distributed control algorithm for removing and/or controlling wrinkle state while the fabric is in motion.

- A method was introduced for handling flexible material as a planar entity that can be maintained (eliminate wrinkles) or altered (generate wrinkles) using exclusively planar actuation to control the out-of-plane dimension.

- A handoff protocol was devised, implemented, and demonstrated for the purpose of sharing state information between neighboring independent environment nodes (eg. vision systems) to allow material to seamlessly transition between the nodes, underpinning the modularity and scalability of the overall system.

- The concept of the unactuated or passive robot traversing through an energetic or actuated environment that has its own intelligence from communication and interaction was introduced and practically demonstrated.

- Each of these contributions were combined into a comprehensive architecture of vision, actuation, and control to generate a novel, scalable, fully functional distributed actuation based fabric transportation system directly transferrable to industry applications.

REFERENCES

[1] U.S. Bureau of Labor Statistics, *All Employees: Manufacturing*, 2018.

[2] ——, *Manufacturing Sector: Real Output*, 2018.

[3] W. J. Book, S. L. Dickerson, and J. D. Huggins, "Conveyance system that transports fabric," pat., US Patent 8,997,670, 2010.

[4] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen, "Multi-sensor surface analysis for robotic ironing," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, Institute of Electrical and Electronics Engineers Inc., 2016, pp. 5670–5676.

[5] Y. Li, C. F. Chen, and P. K. Allen, "Recognition of deformable object category and pose," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014, pp. 5558–5564, ISBN: 9781479936847.

[6] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. Allen, "Folding Deformable Objects using Predictive Simulation and Trajectory Optimization," 2015. arXiv: 1512.06922.

[7] Y. Li, D. Xu, Y. Yue, Y. Wang, S. F. Chang, E. Grinspun, and P. K. Allen, "Regrasping and unfolding of garments using predictive thin shell modeling," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2015, ISBN: VO -.

[8] D. Estevez, J. G. Victores, S. Morante, and C. Balaguer, "Towards Robotic Garment Folding: A Vision Approach for Fold Detection," in *Proceedings - 2016 International Conference on Autonomous Robot Systems and Competitions, ICARSC 2016*, 2016, pp. 188–192, ISBN: 9781509022557.

[9] J. Schrimpf and L. E. Wetterwald, "Experiments towards automated sewing with a multi-robot system," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012, ISBN: 9781467314039.

[10] J. Schrimpf, "Automated sewing using conveyor belts," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2016, ISBN: 9781509013142.

[11] R. C. Winck, S. Dickerson, W. J. Book, and J. D. Huggins, "A novel approach to fabric control for automated sewing," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, 2009, ISBN: 9781424428533.

[12] J. E. Luntz, W. Messner, and H. Choset, "Parcel manipulation and dynamics with a distributed actuator array: the virtual vehicle," *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 2, 1541–1546 vol.2, 1997.

[13] J. Luntz, W. Messner, and H. Choset, "Stick-Slip Operation of the Modular Distributed Manipulator System,"

[14] J. E. Luntz, W. Messner, and H. Choset, "Distributed Manipulation Using Discrete Actuator Arrays," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 553–583, 2001.

[15] J. Luntz, "Dynamics and Control of Discrete Distributed Manipulation,"

[16] J. E. Luntz, W. Messner, and H. Choset, "Discrete Actuator Array Vectorreld Design for Distributed Manipulation,"

[17] M. W. Koch and R. L. Kashyap, "Using Polygons to Recognize and Locate Partially Occluded Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1987.

[18] A. Śluzek, "Using moment invariants to recognize and locate partially occluded 2D objects," *Pattern Recognition Letters*, 1988.

[19] M. W. Koch and R. L. Kashyap, "Matching polygon fragments," *Pattern Recognition Letters*, 1989.

[20] F. Krolupper and J. Flusser, "Polygonal shape description for recognition of partially occluded objects," *Pattern Recognition Letters*, vol. 28, pp. 1002–1011, 2007.

[21] O. Marcotte and S. Suri, "Fast matching algorithms for points on a polygon," *30th Annual Symposium on Foundations of Computer Science*, 1989.

[22] J. Huang, "A new model for general polygon matching problems," *Precision Engineering*, 2009.

[23] M. Zhu, K. G. K. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single Image 3D Object Detection and Pose Estimation for Grasping," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[24] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[25] P. Besl and N. McKay, *A Method for Registration of 3-D Shapes*, 1992.

[26] D. A. Simon, "Fast and Accurate Shape-Based Registration," PhD thesis, 1996, p. 216, ISBN: 0591918854.

[27] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," *Eurographics Symposium on Geometry Processing*, 2013.

[28] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, vol. 2001-Janua, IEEE Computer Society, 2001, pp. 145–152.

[29] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 987–993.

[30] K. Motter, B. Domingue, and W. Book, "Realtime Closed-Loop Fabric Positioning via Distributed Actuation and Vision Feedback," in *Proceedings of 2018 ISFA*, 2018.

[31] ——, "Maintaining Rigid Body Motion of Flexible Material via Distributed Actuation," in *Proceedings of 2018 ISFA*, 2018.

[32] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," *In*, vol. 129, pp. 98–11, 1998. arXiv: arXiv:1011.1669v3.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," in *International Journal of Robotics Research*, vol. 30, 2011, pp. 846–894.

[34] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE International Conference on Intelligent Robots and Systems*, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 2997–3004, ISBN: 9781479969340.